Higher-Order Subtyping

Martin Steffen† Benjamin Pierce‡

February 2, 1994

University of Edinburgh
Technical Report ECS-LFCS-94-280
and
IMMD VII, Universität Erlangen-Nürnberg
Interner Bericht IMMD7-01/94

Revised version with Fun subtyping

Abstract

System $F^\omega_\Xi$ is an extension with subtyping of Girard’s higher-order polymorphic $\lambda$-calculus. We develop the fundamental metatheory of this calculus: decidability of $\beta$-conversion on well-kindred types, elimination of the “cut-rule” of transitivity from the subtype relation, and the soundness, completeness, and termination of algorithms for subtyping and typechecking.

Keywords: lambda-calculus, type systems, subtyping, polymorphism, bounded quantification, typechecking.

1 Introduction

Since the early 1980’s, increasing attention in the programming language community has been devoted to formal models for statically typed object-oriented languages. Cardelli [Car84] observed that refinement of object interfaces can be modeled by records and a simple form of subtyping. To account for the types of message-sending operations, Cardelli and Wegner [CW85] introduced bounded quantification. Fully capturing the object model of languages like Smalltalk required one further refinement, the extension to calculi with higher-order polymorphism, to deal properly with the interaction between subtyping and object encapsulation.

A number of typed object models have been given in this general setting.¹ Cook, Canning, Hill, Olthoff, and Mitchell [CCH⁺89, CHC90] proposed a variant called $F$-bounded quantification, which was used by Bruce [Bru93] to give the first full account of static typing for Smalltalk-style objects. Pierce and Turner [PT93a, PT93b] gave a similar model using existential types instead of recursive types to capture object encapsulation, effectively working within pure $F^\omega_\Xi$. These two approaches were generalized by Hofmann and Pierce [HP94] to an abstract, axiomatic presentation of objects and subtyping.

¹Models for various object-oriented features have also been given using different techniques [MHF93, Aha93, CGL92, Cas92, Car92, AC94b, AC94a, Mit90, Wan87, Wan88, Wan89, etc.].
The core calculus underlying all of these models is an extension of Girard's higher-order polymorphic λ-calculus, System $F^\omega$, with subtyping. Many of the ideas behind this system, called $F^\omega$ ("F-omega-sub"), are due to Cardelli, particularly to his 1988 paper, "Structural Subtyping and the Notion of Power Type" [Car88]; the extension of the subtype relation to type operators was developed by Cardelli and Mitchell [Car90, Mit90, BM92]. An early semantic model was given by Cardelli and Longo [CL91] using partial equivalence relations. Compagnoni and Pierce [CP93] gave a model for an extension of $F^\omega$ with intersection types. A more powerful model including recursive types was given by Bruce and Mitchell [BM92].

The second order fragment of $F^\omega$, called $F_\leq$, has been studied in detail, yielding both positive [Mar88, BL90, BCGS91, Ghe90, CG92, CG91, CMM91, Ghe93b] and some surprising negative results [Ghe93a, Ghe93b, GP92, Pie92] including undecidability. Decidable variants of $F_\leq$ have been proposed [CW85, KS92, CP94] (our formulation of $F^\omega$ is based on [CW85]). But for the full $\omega$-order calculus, next to nothing is known.2

The analysis of $F^\omega_\leq$ is significantly more challenging than that of $F_\leq$, principally because $F^\omega_\leq$ introduces a rule of conversion guaranteeing that β-convergent types occupy the same equivalence class in the subtype relation. This rule interacts with the rule of transitivity, requiring a substantial generalization of the standard cut-elimination argument — a key step in the proof of decidability, where uses of transitivity are restricted to a well-behaved form. Another significant difficulty is showing the termination of the final algorithm; in decidable variants on $F_\leq$, this is fairly straightforward; here, the proof depends on the strong normalization of an unusual notion of reduction on types, in which type variables may be replaced by their upper bounds from the context.

Our goal is to establish fundamental meta-theoretic results for $F^\omega_\leq$, leading up to sound and complete algorithms for checking the subtyping and typing relations. We begin in Section 2 by introducing $F^\omega_\leq$. Sections 3, 4, and 5 develop preliminary results needed in Section 6, the core of the paper, where the decidability of subtyping is proved. Section 7 extends the analysis to the decidability of typing.

In the technical development, we sometimes elide routine proofs. Those not shown are straightforward when performed in the order given.

## 2 Definition of $F^\omega_\leq$

Girard's System $F^\omega$ [Gir72] is a typed λ-calculus with higher-order polymorphism. Besides the term abstraction ($\text{fun } (a:T) t$) and application ($f a$) of the simply typed λ-calculus [Chu40] and the type abstraction ($\text{fun } (A:K) T$) and application ($T t$) of the second-order polymorphic λ-calculus [Gir72, Rey74], it includes the possibility of abstraction ($\text{Fun } (A:K) T$) and application ($T U$) within type expressions. To guarantee the well-formedness of applications within types, an extra level of kinds is introduced: the kind * classifies ordinary types (which are inhabited by terms), while kinds of the form $K_1 \rightarrow K_2$ classify type operators: functions mapping types of kind $K_1$ to types of kind $K_2$. The basic typing judgement for $F^\omega_\leq$ is $\Gamma \vdash t : T$, read "term $t$ has type $T$ in context $\Gamma$," where $\Gamma$ records the type of each free variable $x$ and the kind of each free type variable $A$.

To extend $F^\omega$ with subtyping, we introduce an ordering $S \leq T$ on the elements of each kind $K$. The declaration of each type variable $A$ in $\Gamma$ is extended with an upper bound, written $A \leq T$, which constrains $A$ to range only over subtypes of $T$ in the appropriate kind. To allow new constraints of this form to be introduced

---

2 Compagnoni [Com04] has independently achieved some results closely related to ours. After early joint work on the formulation of $F^\omega_\leq$ (c.f. [CP93]), vagaries of geography led our efforts onto separate, but parallel, tracks. Leaving aside inessential technical differences — our "reducing" system in Section 6.2 performs reduction in the premises of the rules, whereas analogous "normalizing system" assumes that the conclusion is already normalized; she proves Church-Rosser by marking redices, while we adapt Tait and Martin-Löf's method of parallel reduction; etc. — the two proofs are broadly similar in structure. The major differences are as follows. 1) Compagnoni's results are for a more powerful system, $F^\omega_\leq$, that includes intersection types in addition to the machinery of $F^\omega_\leq$. 2) Our development addresses the decidability of typechecking in addition to subtyping. 3) Compagnoni's version of the crucial substitution lemma (our Lemma 6.4.3) is phrased more cleverly and its proof requires a less intricate analysis. 4) Our proof of the termination of the subtyping algorithm is based on showing strong normalization for an extended notion of reduction in which type variables may be replaced by their upper bounds from the context; the argument is rather difficult, but introduces techniques that may be useful in a broader range of calculi including, for example, type abbreviations. Compagnoni uses a more direct term rewriting technique.
into the context, we extend the universal quantifier, $\forall(A:K)U$, to a bounded quantifier $\forall(A\leq T)U$.\footnote{We could also extend type operators $Fun(A:K)U$ to bounded operators $Fun(A\leq T)U$, but it is not clear that this refinement would be useful in practice, and it would complicate the metatheory, since we would then need to introduce a subtyping relation.}

To ensure that the new system can still type all the terms of $F^=\omega$, we assume that the subtype relation in every kind $K$ has a maximal element $\top_K(K)$. The assumption $A\leq \top_K(K)$ replaces $A:K$.

For kinds of the form $K_1\to K_2$, the subtype relation is just the pointwise extension of subtyping for $K_2$: a function $S \in K_1\to K_2$ is smaller than a function $T \in K_1\to K_2$ if $S U \leq T U$ for every $U \in K_1$.

At the base kind $\ast$, the subtype relation also includes rules for the type constructors $T_1\to T_2$ and $\forall(A\leq T_1)T_2$. The rule for arrow types embodies the familiar contravariant/covariant inclusion of function spaces:

$\Gamma \vdash T_1 \leq S_1 \quad \Gamma \vdash T_2 \leq S_2$  
$\Gamma \vdash S_1 \to S_2 \leq T_1 \to T_2$

Intuitively, a function $f$ whose results inhabit $S_2$ whenever its arguments inhabit $S_1$ may safely be substituted for a function in $T_1 \to T_2$, provided that any element of $T_1$ that might be given as an argument to $f$ can safely be used as an element of $S_1$ and that $f$'s result, an element of $S_2$, can be used in place of the expected $T_2$.

The subtyping rule for bounded quantifiers is equally simple:

$\Gamma, A \leq U \vdash S_2 \leq T_2$  
$\Gamma \vdash \forall(A\leq U)S_2 \leq \forall(A\leq U)T_2$

That is, a polymorphic function $f \in \forall(A\leq U)S_2$ can be used in a context that expects an element of $\forall(A\leq U)T_2$, provided that, for each legal argument type $T$, the value of $f$ at $T$ can safely be used as an element of $T_2$.

It would be semantically sensible to refine the right-hand premise of this rule so that it only requires $S_2 \leq T_2$ when $A$ is constrained to the common part of their domain:

$\Gamma \vdash T_1 \leq S_1 \quad \Gamma, A \leq \top(K) \vdash S_2 \leq T_2$  
$\Gamma \vdash \forall(A\leq S_1)S_2 \leq \forall(A\leq \top_K)T_2$

This, indeed, is the form in which the rule appears in most presentations of second-order bounded quantification (c.f. [CP94] for a survey). However, the extra flexibility offered by this refinement does not seem to be useful in practice and it is very costly: this rule is responsible for the failure of a number of important proof-theoretic properties in standard formulations of $F^=\leq$ [Ghe93a, Ghe93b, GP92, Pie92], including decidability of subtyping.

Another variant of the quantifier subtyping rule allows the bounds to differ but requires that the bodies be in the subtype relation under the trivial assumption on the bound variable:

$\Gamma \vdash T_1 \leq S_1 \quad \Gamma, A \leq \top(K) \vdash S_2 \leq T_2$  
$\Gamma \vdash \forall(A\leq S_1)S_2 \leq \forall(A\leq \top_K)T_2$

(where $K$ is the kind of $S_1$ and $T_1$). Indeed, an earlier draft of this paper used this rule instead of the equal-bounds variant. All of the results about subtyping hold for both systems (with nearly identical proofs). But the algorithm for synthesizing minimal types works only for the equal-bounds rule, and not (as we erroneously claimed) for the top-rule. Indeed, the top-rule actually destroys the minimal typing property!

For example, in the context

$\Gamma = \forall(X \leq \top)\ y$  
the term

$\epsilon \equiv \text{fun}\ X (y \leq X) \text{fun}\ (x:X)\ x$

has both of the types

$\forall(X \leq Y)X \to X$  
$\forall(X \leq Y)X \to Y$,

but these types are incomparable (using the top-rule) and have no common lower bound. We are indebted to Giorgio Ghelli for this example.

\footnote{Again, richer definitions of operator subtyping are possible: for example, we might allow monotone subtyping, antimonotone subtyping, etc. [Gar90]. This extension does seem useful in practice (e.g. [HP94]), but its algorithmic implications are unclear.}
2.1 Syntax

The kinds, types, terms, and contexts of $F^\omega_\Sigma$ are defined by the following abstract grammar:

$$
K ::= \,* \quad \text{kind of types} \\
| \,K\rightarrow K \quad \text{kind of type operators}
$$

$$
T ::= \,A \quad \text{type variable} \\
| \,Fun\,(A;\,K)\,T \quad \text{type operator} \\
| \,T\,T \quad \text{application of a type operator} \\
| \,Top\,(K) \quad \text{maximal type} \\
| \,T\rightarrow T \quad \text{function type} \\
| \,\forall (K)\,T \quad \text{universally quantified type}
$$

$$
t ::= \,x \quad \text{variable} \\
| \,\text{fun}\,(x:T)\,t \quad \text{abstraction} \\
| \,t\,t \quad \text{application} \\
| \,\text{fun}\,(A\leq T)\,t \quad \text{type abstraction} \\
| \,t\,T \quad \text{type application}
$$

$$
\Gamma ::= \,\cdot \quad \text{empty context} \\
| \,\Gamma,\,x:T \quad \text{variable binding} \\
| \,\Gamma,\,\forall (A)\,T \quad \text{type variable binding with bound}
$$

The inference rules that follow define sets of derivable statements of the following forms:

$$
\vdash \Gamma \; ok \quad \Gamma \text{ is a well-formed context} \\
\vdash \Gamma \; T \in K \quad \text{type } T \text{ has kind } K \text{ in context } \Gamma \\
\vdash \Gamma \; S \leq T \quad S \text{ is a subtype of } T \text{ in } \Gamma \\
\vdash \Gamma \; t \in T \quad \text{term } t \text{ has type } T \text{ in } \Gamma.
$$

Terms, types, contexts, and statements that differ only in the names of bound variables are regarded as identical.

2.2 Contexts and Kinding

Well-formed contexts are constructed from the empty context by adding well-kindied type and term variable declarations.

$$
\vdash \cdot \; ok \quad \text{(C-EMPTY)} \\
\Gamma \vdash T \in K \quad \forall A \not\in \text{dom}(\Gamma) \\
\vdash \Gamma,\,\forall (A)\,T \; ok \quad \text{(C-TVAR)} \\
\Gamma \vdash T \in \; x \quad \forall x \not\in \text{dom}(\Gamma) \\
\vdash \Gamma,\,x:T \; ok \quad \text{(C-VAR)}
$$

Since the side conditions guarantee that the bindings in well-formed contexts are always for distinct variables, we often consider them as finite functions from variables to types; for example, the upper bound of $A$ in $\Gamma$ is written $\Gamma(A)$. We write $\text{dom}(\Gamma)$ for the set of term and type variables bound by $\Gamma$. If $\Gamma$ is a prefix of $\Gamma'$, we say that $\Gamma'$ is an extension of $\Gamma$.

The definition of the kinding relation is standard. Type variables have the same kind as their upper bounds; abstraction and application provide introduction and elimination forms for arrow-kinds; $\text{Top}(K)$ has kind $K$; arrow- and All-types are well-kindied if their components are. We maintain the invariant that kinding statements are only derivable in well-formed contexts.
\[
\begin{align*}
\Gamma \vdash \Gamma(A) \in K & \quad \text{(K-TVar)} \\
\Gamma \vdash A \in K & \\
\Gamma, A \leq \text{Top}(K_1) \vdash T \in K_2 & \quad \text{(K-Arrow-I)} \\
\Gamma \vdash \text{Fun}(A:K_1)T \in K_1 \rightarrow K_2 & \\
\Gamma \vdash S \in K_1 \rightarrow K_2 & \quad \text{(K-Arrow-E)} \\
\Gamma \vdash ST \in K_2 & \\
\vdash \Gamma \text{ok} & \quad \text{(K-Top)} \\
\Gamma \vdash T_1 \in \ast & \quad \text{(K-Arrow)} \\
\Gamma \vdash T_2 \in \ast & \\
\Gamma, A \leq T_1 \vdash T_2 \in \ast & \quad \text{(K-All)} \\
\Gamma \vdash \text{All}(A \leq T_1)T_2 \in \ast &
\end{align*}
\]

(The kinding and context well-formedness judgements are mutually recursive, but the two main judgement forms — subtyping and typing — only depend non-recursively on other judgements.)

### 2.3 Conversion

The presence of abstractions and applications in type expressions leads us to consider conversion within types. For technical convenience, we use a slight extension of the standard $\beta$-conversion relation: in addition to reductions of the usual form \((\text{Fun}(A:K_1)T) \ U \longrightarrow^\beta [U/A]T\) we allow reductions of the form \(\text{Top}(K_1 \rightarrow K_2) \ T \longrightarrow^\top \text{Top}(K_2)\), which relate the maximal elements of different kinds. We could achieve the same effect by extending the rule S-Top below, but this way seems cleaner.

#### 2.3.1 Definition [\(\beta T\)-reduction]:

One-step $\beta T$-reduction is the smallest relation on types closed under the following rules:

\[
\begin{align*}
\text{Top}(K_1 \rightarrow K_2) & \rightarrow_{\beta T} \text{Top}(K_2) \\
S & \rightarrow_{\beta T} S' \\
ST & \rightarrow_{\beta T} S'T \\
S & \rightarrow_{\beta T} S' \\
(S \rightarrow T) & \rightarrow_{\beta T} (S' \rightarrow T) \\
S & \rightarrow_{\beta T} S' \\
\text{All}(A \leq S | T) & \rightarrow_{\beta T} \text{All}(A \leq S' | T) \\
S & \rightarrow_{\beta T} S' \\
\text{Fun}(A:K)S & \rightarrow_{\beta T} \text{Fun}(A:K)S'
\end{align*}
\]

The many-step $\beta T$-reduction relation \(\longrightarrow^\ast_{\beta T}\) is the reflexive and transitive closure of one-step reduction; \(=_{\beta T}\) is its reflexive, transitive, and symmetric closure. When \(T\) has a normal form (it will necessarily be unique), we denote it by \(T'\). Reduction to $\beta T$-normal form is written \(\longrightarrow^\ast_{\beta T}\).
2.4 Subtyping

The $F_\omega^\rightarrow$ subtyping relation $\Gamma \vdash S \leq T$ is a straightforward extension of the subtyping relation of $F_\omega$ [CW85, CG92, CP94]. We start by stipulating that $\beta\gamma$-convertible types always lie in the same equivalence class in the subtype ordering (S-Conv), and that the subtype relation at every kind is reflexive (R-Refl) and transitive (R-Trans). Type assumptions from the context may be used as axioms (S-Var). $\text{Top}(K)$ is maximal in the ordering for kind $K$ (S-Top). Type operators (S-Abs) and applications (S-App) are subtyped pointwise. Arrow- and All- types have the rules discussed above (S-Arrow and S-All).

In several places in the definition, we add premises to ensure that a proper kinding discipline is respected (e.g., $S \leq \text{Top}(K)$ only when $S \in K$, etc.). But for readability, these are kept to a minimum: we maintain the invariant that whenever the conclusion of a subtyping judgement is well-kind ed, the types on the right- and left-hand sides of the $\leq$ will have the same kind and all of the subderivations will be similarly well behaved.

\[
\begin{array}{ccc}
\Gamma \vdash S \leq U & \Gamma \vdash U \in K & U \Rightarrow T
\\
\Gamma \vdash S \leq T & \Gamma \vdash T \leq T
\\
\Gamma \vdash S \leq U & \Gamma \vdash U \leq T & \Gamma \vdash U \in K
\\
\Gamma \vdash S \leq T
\\
\Gamma \vdash A \leq \Gamma(A)
\\
\Gamma \vdash S \in K
\\
\Gamma \vdash S \leq \text{Top}(K)
\\
\Gamma, A \leq \text{Top}(K) \vdash S \leq T
\\
\Gamma \vdash \text{Fun}(A;K)S \leq \text{Fun}(A;K)T
\\
\Gamma \vdash S \leq T
\\
\Gamma \vdash S \leq T
\\
\Gamma \vdash T_1 \leq S_1 & \Gamma \vdash S_2 \leq T_2
\\
\Gamma \vdash S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2
\\
\Gamma, A \leq U \vdash S_2 \leq T_2
\\
\Gamma \vdash \text{All}(A \leq U)S_2 \leq \text{All}(A \leq U)T_2
\end{array}
\]

(S-Conv) (S-Refl) (S-Trans) (S-Var) (S-Abs) (S-Top) (S-App) (S-Var) (S-Arrow) (S-All)

2.5 Typing

The typing relation $\Gamma \vdash \cdot \in T$ is standard [CW85, CG92], modulo a few extra kinding assumptions. The rule T-SUBSUMPTION captures the intended interpretation of subtyping as “safe substitutability.” The others are straightforward extensions of the arrow- and All-introduction and -elimination rules of pure $F^\rightarrow$.

\[
\begin{array}{cccc}
\Gamma \vdash s \in S & \Gamma \vdash T \in \cdot & \Gamma \vdash S \leq T
\\
\Gamma \vdash s \in T
\\
\vdash \Gamma \text{ok}
\\
\Gamma \vdash x \in \Gamma(x)
\\
\Gamma, x:T_1 \vdash t \in T_2
\\
\Gamma \vdash \text{fun}(x:T_1)t \in T_1 \rightarrow T_2
\\
\Gamma \vdash f \in T_1 \rightarrow T_2
\\
\Gamma, A \leq T_1 \vdash t \in T_2
\\
\Gamma \vdash \text{fun}(A \leq T_1)t \in \text{All}(A \leq T_1)T_2
\\
\Gamma \vdash f \in \text{All}(A \leq T_1)T_2
\\
\Gamma \vdash S \in K
\\
\Gamma \vdash S \leq T_1
\\
\Gamma \vdash f S \in [S/A]T_2
\end{array}
\]

(T-Subsumption) (T-Var) (T-Arrow-I) (T-Arrow-E) (T-All-I) (T-All-E)
3 Properties of Reduction

We now pause to establish some technical properties of the reduction relation and to define an auxiliary notion of parallel reduction that will simplify some of the inductive arguments in later sections. The main result of this section is the Church-Rosser property, by a straightforward adaptation of Tait and Martin-Löf’s proof for ordinary \( \beta \)-reduction (c.f. [Bar84]).

3.1 Definition [Parallel reduction]: Single-step parallel reduction is the least relation closed under the following rules:

\[
\begin{align*}
\text{Top}(K_1 \rightarrow K_2) & \xrightarrow{\beta_T} \text{Top}(K_2) \\
S & \xrightarrow{\beta_T} S' & T & \xrightarrow{\beta_T} T' \\
ST & \xrightarrow{\beta_T} S'T' \\
S \rightarrow T & \xrightarrow{\beta_T} S' \rightarrow T' \\
T & \xrightarrow{\beta_T} T
\end{align*}
\]

Ordinary single-step reduction is a subrelation of single-step parallel reduction, which is a subrelation of multi-step ordinary reduction. The reflexive, transitive closures of the two relations coincide:

3.2 Fact:

1. \( \rightarrow_{\beta_T} \supset \xrightarrow{\beta_T} \)
2. \( \xrightarrow{\beta_T} \supset \rightarrow_{\beta_T} \)
3. \( \rightarrow^*_{\beta_T} = \rightarrow_{\beta_T} \)

Substitution commutes with parallel and multi-step reduction:

3.3 Lemma:

1. If \( S \xrightarrow{\beta_T} S' \) and \( T \xrightarrow{\beta_T} T' \) then \( [T/A]S \xrightarrow{\beta_T} [T'/A]S' \).
2. If \( S \xrightarrow{\beta_T} S' \) and \( T \xrightarrow{\beta_T} T' \) then \( [T/A]S \xrightarrow{\beta_T} [T'/A]S' \).
3. If \( S \xrightarrow{\beta_T} S' \) and \( T \xrightarrow{\beta_T} T' \) then \( [T/A]S \xrightarrow{\beta_T} [T'/A]S' \).

In the proof of 3.3, we need the following property of substitution:

3.4 Fact: If \( A \neq A' \) and \( A' \not\in \text{FV}(S) \), then \( [S/A][T/A']U = [[S/A]T/A'][[S/A]U] \).

One useful consequence of 3.3(3) is that if an expression with an outermost redex has a reduction path in which this redex is reduced at some point, then this reduction can be performed first without changing the result:

3.5 Corollary [Outermost reduction]: If \( (\text{Fun}(A;K)S)T \xrightarrow{\beta_T} U \), where \( U \neq (\text{Fun}(A;K)S')T' \) with \( S \xrightarrow{\beta_T} S' \), and \( T \xrightarrow{\beta_T} T' \), then \( [T/A]S \xrightarrow{\beta_T} U \).
3.6 Lemma [Diamond property for \(\beta_T\)]: For all types \(S, S_1, \) and \(S_2\) with \(S \rightarrow^\beta_T S_1\) and \(S \rightarrow^\beta_T S_2\), there is a type \(S_3\) such that \(S_1 \rightarrow^\beta_T S_3\) and \(S_2 \rightarrow^\beta_T S_3\).

\[
\begin{array}{c}
S \\
\downarrow \beta_T \\
S_1 \\
\downarrow \beta_T \\
S_2 \\
\downarrow \beta_T \\
S_3
\end{array}
\]

Proof: Straightforward extension of the standard argument (c.f. [Bar84]).

3.7 Corollary [Church-Rosser for \(\beta_T\)]: For all types \(S, S_1, \) and \(S_2\) with \(S \rightarrow^\beta_T S_1\) and \(S \rightarrow^\beta_T S_2\), there is a type \(S_3\) such that \(S_1 \rightarrow^\beta_T S_3\) and \(S_2 \rightarrow^\beta_T S_3\).

The proof of strong normalization for \(\rightarrow^\beta_T\) has to be deferred until after we have studied the properties of the kinding system, since the proof of normalization requires that the types involved be well kinded.

One more property of \(\rightarrow^\beta_T\) will be needed for the induction in the proof of Lemma 6.4.8.

3.8 Lemma [\(\rightarrow^\beta_T\) and \(\rightarrow^\beta_T\)]: If \(S \rightarrow^\beta_T S_1\) and \(S \rightarrow^\beta_T S_2\), then there is an \(S_3\) with \(S_1 \rightarrow^\beta_T S_3\) and \(S_2 \rightarrow^\beta_T S_3\).

\[
\begin{array}{c}
S \\
\downarrow \beta_T \\
S_1 \\
\downarrow \beta_T \\
S_2 \\
\downarrow \beta_T \\
S_3
\end{array}
\]

Proof: By induction on the length of \(S \rightarrow^\beta_T S_2\), using 3.2(1) and Church-Rosser.

4 Kinding

Next, we state some basic technical properties of the context well-formedness and kinding judgements.

4.1 Lemma [Generation of contexts]:

1. If \(\Gamma \vdash \emptyset\), then:
   - (a) \(\Gamma = \emptyset\); or
   - (b) \(\Gamma = \Gamma_1, \mathcal{A}: T\), with \(\Gamma_1 \vdash \emptyset\) and \(\Gamma_1 \vdash T \in \mathcal{A}\) as subderivations; or
   - (c) \(\Gamma = \Gamma_1, A \leq T\), with \(\Gamma_1 \vdash \emptyset\) and \(\Gamma_1 \vdash T \in K\) for some \(K\) as subderivations.

2. If \(\Gamma \vdash S \in K\), then \(\Gamma \vdash \emptyset\) as a subderivation.

4.2 Lemma [Generation of types]:

1. If \(\Gamma \vdash A \in K\), then \(\Gamma \vdash \Gamma(A) \in K\).

2. If \(\Gamma \vdash \text{Fun}(A; K_1) \in K\), then, for some \(K_2\), we have \(\Gamma, A \leq \text{Top}(K_1) \vdash T \in K_2\) and \(K = K_1 \rightarrow K_2\).

3. If \(\Gamma \vdash S \in K\), then, for some \(K'\), we have \(\Gamma \vdash S \in K' \rightarrow K\) and \(\Gamma \vdash T \in K'\).

4. If \(\Gamma \vdash S \rightarrow T \in K\), then \(K = \star\) and \(\Gamma \vdash S, T \in \star\).

5. If \(\Gamma \vdash \text{Top}(K) \in K'\), then \(K = K'\).
6. If \( \Gamma \vdash \text{All}(A \leq S)T \in K \), then \( K = \emptyset \) and \( \Gamma, A \leq S \vdash T \in K \).

Moreover, the implied derivations are all subderivations of the originals.

We prove the decidability of the kinding system by showing that it is equivalent to a different system whose decidability is obvious.

4.3 Definition [Algorithmic kinding]: The algorithmic kinding relation \( \Gamma \vdash \_A T \in K \) is the least relation closed under the kinding rules, where instead of rule K-TVAR we use the following:

\[
\Gamma \vdash \_A T \in K \quad \vdash \_A \Gamma \leq T, \Gamma \_2 \text{ ok} \quad \Gamma \vdash A \in K
\]

(K-TVAR')

The algorithmic kinding relation \( \vdash \_A \) ok is defined as before, using algorithmic kinding.

4.4 Lemma [Context strengthening for algorithmic kinding]:

1. If \( \Gamma, A \leq S, \Gamma \_2 \vdash A T \in K \) and \( A \) is not free in \( \Gamma \_2 \) or in \( T \), then \( \Gamma \_1, \Gamma \_2 \vdash A T \in K \).
2. If \( \vdash \_A \Gamma \_1, A \leq S, \Gamma \_2 \text{ ok} \) and \( A \) is not free in \( \Gamma \_2 \), then \( \vdash \_A \Gamma \_1, \Gamma \_2 \text{ ok} \).
3. If \( \Gamma \_1, x : S, \Gamma \_2 \vdash A T \in K \), then \( \Gamma \_1, \Gamma \_2 \vdash A T \in K \).
4. If \( \vdash \_A \Gamma \_1, x : S, \Gamma \_2 \text{ ok} \), then \( \vdash \_A \Gamma \_1, \Gamma \_2 \text{ ok} \).

4.5 Lemma [Decidability of kinding]: The relations \( \vdash \_A \text{ ok} \) and \( \vdash S \in K \) are decidable.

Proof: It is easy to prove by induction that the two kinding systems and the two definitions of context well-formedness are equivalent. In each direction, we only have to consider the rule for variables, since all other rules coincide.

Case K-TVAR: \( \vdash \_A (A) \in K \)

By Lemma 4.1(2) \( \vdash \_A \Gamma \_1, A \leq T, \Gamma \_2 \text{ ok} \) as subderivation. So by the induction hypothesis, \( \vdash \_A \Gamma \_1, A \leq T, \Gamma \_2 \text{ ok} \) and \( \vdash \_A \Gamma \_1, A \leq T, \Gamma \_2 \vdash A \in K \). Repeated application of Lemma 4.4 yields \( \vdash \_A \Gamma \_1, A \leq T \in K \).

Case K-TVAR': \( \vdash \_A T \in K \) and \( \vdash \_A \Gamma \_1, A \leq T, \Gamma \_2 \text{ ok} \)

By the induction hypothesis, \( \vdash A T \in K \) and \( \vdash \_A \Gamma \_1, A \leq T, \Gamma \_2 \text{ ok} \), so the result follows by K-TVAR and weakening.

Now, the algorithm obtained by reading the algorithmic kinding rules from bottom to top as Horn clauses always terminates, since in each step the total number of characters in the conclusion is greater than the number of characters in any of the premises. Since the systems are equivalent, \( \vdash S : K \) is also decidable. \( \square \)

4.6 Lemma [Uniqueness of kinding]: If \( \vdash S \in K \) and \( \vdash \text{All}(S)K' \), then \( K = K' \).

This justifies the following notation:

4.7 Definition: The unique kind of a well-kindled type \( S \) in a context \( \Gamma \) is written \( \text{Kind}_T(S) \).

4.8 Lemma [Transposition and weakening for kinding]: Suppose that \( T' \) is a well-formed extension of \( \Gamma \_1, A \leq T, A \leq S, \Gamma \_2 \). If \( \Gamma \_1, A \leq S, A \leq T, \Gamma \_2 \vdash U \in K \) and \( A \notin FV(T) \), then \( \Gamma \vdash U \in K \).

4.9 Lemma [Context update for kinding]: If \( \Gamma \_1, A \leq S, \Gamma \_2 \vdash T \in K \) and \( \Gamma \_1 \vdash S, S \_2 \in K' \), then \( \Gamma \_1, A \leq S, \Gamma \_2 \vdash T \in K \).

4.10 Lemma [Top reduction]: If \( \vdash \text{Top}(K) T \_1 \ldots T \_n \in K' \), then \( \text{Top}(K) T \_1 \ldots T \_n \rightarrow^*_\beta \text{Top}(K') \).

4.11 Lemma [Kinding and substitution]: Suppose \( \Gamma \vdash T \in K' \).

1. If \( \vdash \Gamma \_1, A \leq \text{Top}(K'), \Gamma \_2 \text{ ok} \), then \( \vdash \_A \Gamma \_1, [T / A] \Gamma \_2 \text{ ok} \).
2. If \( \vdash \Gamma \_1, A \leq \text{Top}(K'), \Gamma \_2 \vdash S \in K \), then \( \vdash \_A [T / A] \Gamma \_2 \vdash [T / A] S \in K \).
Proof: Both parts are proved simultaneously by induction on derivations.

4.12 Lemma [Subject reduction for types and contexts]:

1. If $\Gamma \vdash S \in K$ and $S \rightarrow^\beta T$ then $\Gamma \vdash T \in K$.
2. If $\Gamma \vdash S \in K$ and $\Gamma \rightarrow^\beta \Gamma'$, then $\Gamma' \vdash S \in K$.

4.13 Corollary [Kind invariance under conversion]: If $S \equiv^\beta T$, where $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$, then $K_S = K_T$.

5 Strong Normalization of Types

We shall often need the fact that $\beta$-reduction is strongly normalizing for well-kindred types. In fact, we prove the strong normalization of a more general reduction relation, called $\beta\Gamma$-reduction, which will be used to prove the termination of the subtyping and typing algorithms. Besides the usual $\beta$ and $\Gamma$ reductions, we allow variables to be replaced by their upper bounds from the context. $\Gamma$-reduction is reminiscent of the common operation in type checker and proof checker implementations of replacing a type definition by its expansion (c.f. [SP93]).

We begin by proving the strong normalization of $\beta\Gamma$-reduction, using by a straightforward translation argument.

5.1 Definition: Define a family of types $T_K$, one for each kind $K$, as follows:

$$
T_s = \text{All}(A: \times) A
$$

$$
T_{K_1 - K_2} = \text{Fun}(A:K_1) T_{K_2}
$$

Note that $\Gamma \vdash T_K \in K$ for each $K$.

5.2 Lemma [Strong $\beta\Gamma$-normalization]: Suppose $\Gamma \vdash S \in K$. Then there is no infinite $\beta\Gamma$-reduction from $S$.

Proof: Define a translation function $F$ mapping $F^\omega_K$ types to $F^\omega$ types:

$$
F(\text{Top}(K)) = T_K
$$

$$
F(\text{All}(A:S) T) = \text{All}(A) F(S) \rightarrow F(T)
$$

$$
F(\text{Fun}(A:K) S) = \text{Fun}(A:K) F(S)
$$

$$
F(S T) = F(S) F(T)
$$

On contexts, $F$ replaces each type variable binding $A:S$ in $\Gamma$ by the kinding assumption $A:K$, where $K$ is the kind of $T$ in $\Gamma$. It is easy to check that if $\Gamma \vdash S \in K$ in $F^\omega_K$, then $F(\Gamma) \vdash F(S) \in K$ in $F^\omega$.

Now, any $\beta\Gamma$-reduction in $F^\omega_K$ from $S$ can be mirrored by a $\beta$-reduction from $F(S)$ of the same length in $F^\omega$. The existence of an infinite $\beta\Gamma$-reduction in $F^\omega_K$ would thus contradict the strong normalization of $F^\omega$ [Gir72, Gal90].

Next, we define the notion of $\Gamma$-reduction and establish some of its basic properties.
5.3 Definition: Single-step $\Gamma$-reduction is the least family of relations closed under:

\[
\begin{align*}
\Gamma(A) & \neq \text{Top}(K) \\
A & \xrightarrow{\Gamma} \Gamma(A) \\
S & \xrightarrow{r} S' \\
ST & \xrightarrow{r} ST' \\
S & \xrightarrow{r} S' \\
(S-T) & \xrightarrow{r} (S'-T) \\
All(A \leq S | T) & \xrightarrow{r} All(A \leq S' | T) \\
\end{align*}
\]

Single-step $\beta$TT-reduction, written $\xrightarrow{\beta \Gamma}$, is the least family of relations closed under these rules and the rules ($\beta$) and (T) of Definition 2.3.1. The corresponding multi-step reductions are defined as usual. Note that in a multi-step $\Gamma$-reduction sequence, the $\Gamma$ at each stage remains the same; $\Gamma$ is only extended "internally," in the course of a single reduction, to keep track of variable bindings in those rules that define reduction under binders.

Note that we are careful to separate All-bound variables, whose bounds may be different from Top and which may thus act as $\Gamma$-redexes, from Fun-bound variables, whose bounds are always Top and which can never be $\Gamma$-reduced.

5.4 Lemma [Strong $\Gamma$-normalization]: If $\Gamma \vdash S \in K$, then there is no infinite $\Gamma$-reduction from $S$.

Proof: We show, by induction on the definition of $\xrightarrow{\Gamma}$, that if $S \xrightarrow{\Gamma} T$ in one step, then $\Gamma \vdash T \in K$ by a shorter derivation. Most cases are straightforward; we list only the ones for type variables and for All-types.

Case: $S = A$ and $T = \Gamma(A)$

By the generation lemma for types, $\Gamma \vdash \Gamma(A) \in K$ by a shorter derivation.

Case: $S = All(A \leq S_1)S_2$ and $T = All(A \leq S_1)S_2$

By the generation lemma for types and contexts and Lemma 4.1, $\Gamma \vdash S_1 \in K_1$, so by the induction hypothesis $\Gamma \vdash S'_1 \in K_1$ by a shorter derivation. Finally $\Gamma \vdash All(A \leq S'_1)S_2 \in \ast$ by K-Att., by a shorter derivation than the original.

Case: $S = All(A \leq S_1)S_2$ and $T = All(A \leq S_1)S_2$

By the generation lemma for types, $A \leq S_1 \subseteq S_2 \subseteq K_1$. The induction hypothesis gives $\Gamma, A \leq S_1 \vdash S'_2 \in K_2$ by a shorter derivation, so by K-Att. we have $\Gamma \vdash All(A \leq S_1)S'_2$ by a shorter derivation than the original.

We shall often use the subject-reduction property silently in what follows, to guarantee that a reduction sequence from a well-kindred term only contains well-kindred terms.

5.5 Lemma [Subject reduction]: If $\Gamma \vdash S \in K$ and $S \xrightarrow{\beta \Gamma} S'$, then $\Gamma \vdash S' \in K$.

Proof: By induction on the length of the reduction $S \xrightarrow{\beta \Gamma} S'$, with an inner induction on the definition of single-step $\beta \Gamma$-reduction.

5.6 Fact:

1. If $All(A \leq T_1)T_2 \xrightarrow{\Gamma} All(A \leq V_1)V_2$, then $T_1 \xrightarrow{r} V_1$ and $T_2 \xrightarrow{r} All(A \leq T_1)V_2$.

2. If $Fun(A:K)T_2 \xrightarrow{\Gamma} Fun(A:K)V_2$, then $T_2 \xrightarrow{r} V_2$.

3. If $T_1 T_2 \xrightarrow{\Gamma} V_1 V_2$, then $T_1 \xrightarrow{r} V_1$ and $T_2 \xrightarrow{r} V_2$. 

11
4. If \((T_1 \to T_2) \to \Gamma \ * (V_1 \to V_2)\), then \(T_1 \to \Gamma \ * V_1\) and \(T_2 \to \Gamma \ * V_2\).

**Proof:** We give the proof in detail for part (1); the rest are similar, but simpler. For part (1), we prove the more refined statement

if \(T_1 \to \Gamma \ * U_1\) and \(T_2 \to \Gamma \ * (\Gamma, A \leq T_1) \ U_2\) and \(\text{All}(A \leq U_1) U_2 \to \Gamma \ * \text{All}(A \leq V_1) V_2\), then

\(T_1 \to \Gamma \ * V_1\) and \(T_2 \to \Gamma \ * (\Gamma, A \leq T_1) \ V_2\),

by induction on the length of \(\alpha\).

**Case:** \(\alpha\) is empty

Immediate.

**Case:** \(\text{All}(A \leq U_1) U_2 \to \Gamma \ * \text{All}(A \leq U_1) U_2 \to \Gamma \ * \text{All}(A \leq V_1) V_2\)

(I.e., \(\alpha\) consists of a single-step reduction \(\alpha_1\) followed by a multi-step reduction \(\alpha_2\), where \(\alpha_1\) replaces a single variable in \(U_1\) by its upper bound to yield \(U_2\).) To apply the induction hypothesis, we need to check that \(U_2 \to \Gamma \ * (\Gamma, A \leq T_1) \ U_2\), which immediately gives \(T_2 \to \Gamma \ * (\Gamma, A \leq T_1) \ U_2\).

But if \(\alpha_1\) replaces an occurrence of \(A\) by \(U_1\) in \(U_2\), i.e. \(U_2 = U_2[A] \to \Gamma, A \leq U_1) \ U_2[U_1]\), then we can build a reduction \(U_2[A] \to \Gamma, A \leq T_1) \ U_2[U_1]\) by replacing this occurrence of \(A\) with \(T_1\) and then using the assumption that \(T_1 \to \Gamma \ * U_1\) (and hence \(T_1 \to \Gamma, A \leq T_1) \ U_1\)) to develop \(T_1\) to \(U_1\) in-place. On the other hand, if \(\alpha_1\) replaces some other variable, then \(U_2 \to \Gamma \ * (\Gamma, A \leq T_1) \ U_2\) is immediate. In both cases, the induction hypothesis then applies, directly yielding the desired result.

**Case:** \(\text{All}(A \leq U_1) U_2 \to \Gamma \ * \text{All}(A \leq U_1) U_2 \to \Gamma \ * \text{All}(A \leq V_1) V_2\)

Here

the induction hypothesis applies directly (since \(T_1 \to \Gamma \ * U_1 \to \Gamma \ U_1\) and we have \(T_2 \to \Gamma \ * (\Gamma, A \leq T_1) \ U_2\) by assumption) to yield the desired result.

5.7 **Lemma [Weak diamond property for \(\Gamma\)-reduction]:**

\[
\begin{array}{c}
T \to \Gamma \ U \\
\downarrow \Upsilon \downarrow \\
\Gamma \ * \Upsilon \\
\downarrow \\
V \to \Gamma \\
\end{array}
\]

**Proof:** By induction on the form of \(T\).

**Case:** \(T = A\)

Then \(V = U = \Gamma (A)\) and we may take \(W = \Gamma (A)\).

**Case:** \(T = \text{All}(A \leq T_1) \ T_2\)

We must find \(W = \text{All}(A \leq W_1) W_2\) such that the required diagram commutes; this will follow from the commutativity of a smaller diagram for \(T_1, U_1, V_1\) and \(W_1\) and another diagram for \(T_2, U_2, V_2\) and \(W_2\). There are three subcases to consider, depending on whether both of the reductions from \(T\) to \(U\) and \(V\) are in \(T_1\), both are in \(T_2\), or one is in \(T_1\) and one in \(T_2\). (Since the last case is symmetric, we may assume without loss of generality that \(T_1\) is reduced to produce \(V\) and \(T_2\) to produce \(U\).)

**Subcase:** \(U = \text{All}(A \leq U_1) T_2\) and \(V = \text{All}(A \leq V_1) T_2\)

Begin by applying the induction hypothesis to \(T_1, U_1, V_1\) and \(W_1\) to yield a common reduct \(W_1\). We must then
show:

\[
\begin{array}{c}
T_1 \xrightarrow{\Gamma} U_1 \\
\downarrow \\
\downarrow \\
T_2 \xrightarrow{\Gamma, A \leq T_1} T_2 \\
\downarrow \\
\downarrow \\
V_1 \xrightarrow{\Gamma, A \leq T_1} \ast \\
\downarrow \\
\downarrow \\
\ast \xrightarrow{\Gamma, A \leq U_1} W_2.
\end{array}
\]

Set \( W_2 = T_2 \) and we are done.

**Subcase:** \( U = All(A \leq T_1) \) \( U_2 \) and \( V = All(A \leq T_1) \) \( V_2 \)

We must find a \( W_2 \) such that

\[
\begin{array}{c}
T_1 \xrightarrow{\Gamma} T_1 \\
\downarrow \\
\downarrow \\
T_2 \xrightarrow{\Gamma, A \leq T_1} U_2 \\
\downarrow \\
\downarrow \\
\downarrow \\
V_1 \xrightarrow{\Gamma, A \leq T_1} V_1 \\
\downarrow \\
\downarrow \\
\ast \xrightarrow{\Gamma, A \leq U_1} W_2.
\end{array}
\]

The existence of such a \( W_2 \) is given by the induction hypothesis.

**Subcase:** \( U = All(A \leq T_1) \) \( U_2 \) and \( V = All(A \leq V_1) \) \( T_2 \)

Set \( W_1 = V_1 \). Then we must show:

\[
\begin{array}{c}
T_1 \xrightarrow{\Gamma} T_1 \\
\downarrow \\
\downarrow \\
T_2 \xrightarrow{\Gamma, A \leq T_1} U_2 \\
\downarrow \\
\downarrow \\
\downarrow \\
V_1 \xrightarrow{\Gamma, A \leq T_1} V_1 \\
\downarrow \\
\downarrow \\
\ast \xrightarrow{\Gamma, A \leq V_1} W_2.
\end{array}
\]

If \( T_2 \xrightarrow{[\Gamma, A \leq T_1]} U_2 \) by a \( \Gamma \)-reduction on some occurrence of \( A \) in \( T_2 \), then we have \( T_2 = T_2[A] \) and \( U_2 = T_2[T_1] \); set \( W_2 = T_2[V_1] \). If \( T_2 \xrightarrow{[\Gamma, A \leq T_1]} U_2 \) by a \( \Gamma \)-reduction on some occurrence of a variable other than \( A \) in \( T_2 \), then we can set \( W_2 = U_2 \), since \( T_2 \xrightarrow{[\Gamma, A \leq V_1]} U_2 \) follows directly from \( T_2 \xrightarrow{[\Gamma, A \leq T_1]} U_2 \) in this case.
Other cases:
Straightforward.

5.8 Lemma [Church-Rosser for $\Gamma$-reduction]:

\[
\begin{array}{c}
T \xrightarrow{*_{\Gamma}} U \\
\Gamma \times \Gamma \\
\times \times \\
V \xrightarrow{*_{\Gamma}} W
\end{array}
\]

**Proof:** By Newman’s Lemma, which states that the weak diamond property and strong normalization together imply Church-Rosser (c.f. [Bar84]).

5.9 Lemma [Substitution commutes with $\Gamma$-reduction]: If $U \xrightarrow{T} *V$ and $S \xrightarrow{\Gamma, A \leq \text{Top}(K)} *T$, then $[U/A]S \xrightarrow{T} *[V/A]T$.

**Proof:** Since $A$’s bound is $\text{Top}(K)$, it is not a $(\Gamma, A \leq \text{Top}(K))$-redex, so $[U/A]S \xrightarrow{T} *[U/A]T$. Then $[U/A]T \xrightarrow{T} *[V/A]T$ by applying the reduction from $U$ to $V$ at each point in $[U/A]T$ where $A$ appeared in $T$.

At this point, we can start proving properties relating $\Gamma$-reduction and $\beta T$-reduction. First, a technical property that handles a key step of the following lemma.

5.10 Lemma: If $T_1 \xrightarrow{\beta T} U_1$ and $T_2 \xrightarrow{\Gamma, A \leq T_1} V_2$, then there is some $W_2$ such that:

\[
\begin{array}{c}
T_2 \\
\Gamma, A \leq T_1 \times \times \\
\times \times \times \\
V_2 \xrightarrow{\beta T} W_2
\end{array}
\]

**Proof:** By induction on the length of the reduction from $T_2$ to $V_2$.

Case: $T_2 = V_2$
Then set $W_2 = T_2$ and we are done.

Case: $T_2 \xrightarrow{*_{\Gamma, A \leq T_1}} V_2'$
Apply the induction hypothesis to find a $W_2'$ satisfying the desired property. We must now show:

\[
\begin{array}{c}
T_2 \\
\Gamma, A \leq T_1 \times \times \\
\times \times \times \\
V_2' \xrightarrow{\beta T} W_2'
\end{array}
\]
If $V_2' \xrightarrow{\beta \cdot [\Gamma, A \leq T_1]} V_2$ by contracting a redex other than $A$, then $V_2' = V_2'[B]$ and $V_2 = V_2'[\Gamma(B)]$. In reducing from $V_2'$ to $W_2'$, this redex may be copied a number of times: $W_2' = W_2'[B][B] \ldots [B]$. Let $W_2'$ be the result of contracting the residuals of this redex in $W_2'$, i.e. $W_2 = W_2'[\Gamma(B)][\Gamma(B)] \ldots [\Gamma(B)]$. Similarly, if $V_2' = V_2'[A]$ and $V_2 = V_2'[\Gamma(T_1)]$; again, let $W_2'$ be the result of contracting the residuals of this redex in $W_2'$. Reduce $V_2 = V_2'[T_1]$ to $W_2 = W_2'[V_1][V_1] \ldots [V_1]$ by $V_2'[T_1] \xrightarrow{\beta \cdot \Gamma} W_2'[T_1][T_1] \ldots [T_1] \xrightarrow{\beta \cdot \Gamma} W_2'[U_1][U_1] \ldots [U_1]$. 

The next lemma establishes a confluence property for $\Gamma$ and $\beta \cdot \Gamma$ reductions. The proof is similar to that of Lemma 5.7. This lemma and Lemma 5.8 jointly handle the crucial step in the strong normalization argument that follows.

**5.11 Lemma [$\beta \cdot \Gamma$-reduction and $\Gamma$-reduction]:**

\[
\begin{array}{c}
T \\
\xrightarrow{\alpha} \\
\xrightarrow{\beta \cdot \Gamma} \\
\xrightarrow{\Gamma \cdot \alpha'} \\
\xrightarrow{\beta \cdot \Gamma \cdot \alpha''} \\
W \end{array}
\]

Moreover, $\alpha'$ has length at least 1.

**Proof:** By induction on $\alpha$.

**Case:** $T = Top(K_1 \rightarrow K_2) T_1 \xrightarrow{\tau} Top(K_2) = U$

Any $\Gamma$-reduction from $T$ must reduce only redexes in $T_1$, so $V$ has the form $Top(K_1 \rightarrow K_2) V_1$. But then $V \xrightarrow{\tau} U$, and we can take $W = U$. Note that $V \xrightarrow{\beta \cdot \Gamma} W$ by a nonempty reduction.

**Case:** $T = (Fun(A;K) T_1) T_2 \xrightarrow{\beta} [T_2/A][T_1] = U$

Any $\Gamma$-reduction from $T$ consists of a number of separate reductions in $T_1$ and $T_2$ by Fact 5.6(2,3), so $V$ has the form $(Fun(A;K) V_1) V_2$, with $T_1 \xrightarrow{\Gamma, A \leq Top(K)} V_1$ and $T_2 \xrightarrow{\Gamma} V_2$. By Lemma 5.9, $[T_2/A][T_1] \xrightarrow{\Gamma} [V_2/A][V_1]$, so we can take $W = [V_2/A][V_1]$. Note that $V \xrightarrow{\beta \cdot \Gamma} W$ by a nonempty reduction.

**Case:** $T = All(A \leq T_1) T_2 \xrightarrow{\beta \cdot \Gamma} All(A \leq U_1) T_2 = U$

By Fact 5.6(1), $V$ has the form $All(A \leq V_1) V_2$, with $T_1 \xrightarrow{\Gamma} [V_2/A][V_1]$ and $T_2 \xrightarrow{\Gamma} V_2$. Apply the induction hypothesis to find a $W_1$ with

\[
\begin{array}{c}
T_1 \\
\xrightarrow{\beta \cdot \Gamma} \\
\xrightarrow{\Gamma \cdot \alpha_1} \\
\xrightarrow{\beta \cdot \Gamma} \\
W_1 \end{array}
\]

where $\alpha_1$ has length at least 1. By Lemma 5.10, there is some $W_2$ such that:

\[
\begin{array}{c}
T_2 \\
\xrightarrow{\beta \cdot \Gamma} \\
\xrightarrow{\Gamma, A \leq U_1} \\
\xrightarrow{\beta \cdot \Gamma} \\
W_2 \end{array}
\]

So $W = All(A \leq U_1) W_2$ has the required property.
Case:

\[
T_2 \xrightarrow{\beta_T} U_2
\]

\[T = \text{All}(A \leq T_1) T_2 \xrightarrow{\beta_T} \text{All}(A \leq T_1) U_2 = U\]

By Lemma 5.6(1), \(V\) has the form \(\text{All}(A \leq V_1) V_2\), with \(T_1 \xrightarrow{\Gamma} V_1\) and \(T_2 \xrightarrow{(\Gamma, A \leq T_1)} V_2\). We must show

\[
\begin{array}{cccccc}
T_1 & \xrightarrow{\beta_T} & T_1 \\
\Gamma \times & \xrightarrow{\beta_T} & U_2 \\
V_1 & \xrightarrow{\beta_T} & V_1 \\
\Gamma, A \leq T_1 & \xrightarrow{\beta_T} & \Gamma, A \leq T_1 \\
V_2 & \xrightarrow{\beta_T} & W_2,
\end{array}
\]

which follows directly from the induction hypothesis.

Other cases:
Similarly, using parts (2) to (4) of 5.6.

With this in hand, we can proceed to the main body of the strong normalization argument. Its two main steps are captured by this lemma and the next one.

5.12 Lemma [\(\beta_T\) postponement]: If \(T \xrightarrow{\beta_T} U \xrightarrow{\Gamma} X \xrightarrow{\beta_T} \ldots\), then there is some \(V_0\) such that \(T \xrightarrow{\Gamma} V_0 \xrightarrow{\beta_T} \ldots\).

For the proof, we need a simple fact:

5.13 Fact: If \(S \xrightarrow{\beta_T} T \xrightarrow{\Gamma} U\), then \(S \xrightarrow{\Gamma} U^\prime\) for some \(U^\prime\). (That is, the redex that is contracted between \(T\) and \(U\) is a residual of a redex already present in \(S\).

Proof: Since \(\beta_T\)-reduction cannot create a \(\Gamma\)-redex, the \(\Gamma\)-redex appearing in \(T\) must be a residual of a \(\Gamma\)-redex already appearing in \(S\).

Proof of Lemma 5.12: By Fact 5.13, there is some \(V_0\) such that:

\[
\begin{array}{cccccc}
T & \xrightarrow{\Gamma} & V_0 \\
\beta_T & \xrightarrow{\beta_T} & U \\
\beta_T^\infty & \xrightarrow{\beta_T^\infty} & \ldots
\end{array}
\]

By Lemma 5.11, there is some \(V_1\) such that:

\[
\begin{array}{cccccc}
T & \xrightarrow{\Gamma} & V_0 \\
\beta_T & \xrightarrow{\beta_T} & U \\
\beta_T^\infty & \xrightarrow{\beta_T^\infty} & \ldots
\end{array}
\]

\]
Since $U \rightarrow_\Gamma X$, we can now apply Lemma 5.8:

\[
\begin{array}{c}
T \rightarrow_\Gamma V_0 \\
\beta T \rightarrow \beta T \Gamma x \\
U \rightarrow_\Gamma V_1 \\
\Gamma \rightarrow_\Gamma \Gamma x \\
X \rightarrow_\Gamma V_2 \\
\beta T \Gamma \Gamma x \rightarrow \beta T \Gamma \Gamma x \infty \\
\vdots
\end{array}
\]

We can continue in this way, applying either 5.8 or 5.11 to successive elements of the infinite reduction beginning from $X$ to obtain an infinite sequence of multi-step $\beta T T$-reductions on the right:

\[
\begin{array}{c}
T \rightarrow_\Gamma V_0 \\
\beta T \rightarrow \beta T \Gamma x \\
U \rightarrow_\Gamma V_1 \\
\Gamma \rightarrow_\Gamma \Gamma x \\
X \rightarrow_\Gamma V_2 \\
\beta T \Gamma \Gamma x \rightarrow \beta T \Gamma \Gamma x \infty \\
U_3 \rightarrow_\Gamma V_3 \\
\beta T \Gamma \Gamma x \rightarrow \beta T \Gamma \Gamma x \infty \\
\vdots \quad \vdots
\end{array}
\]

But the sequence of reductions on the left must contain infinitely many $\beta T$ steps (otherwise it would have an infinite $\Gamma$-tail), so Lemma 5.11 also tells us that infinitely many of the individual multi-step reductions on the right are nonempty. The reduction $T \rightarrow_\Gamma V_0 \rightarrow_\beta T \Gamma x V_1 \cdots \rightarrow_\beta T \Gamma x$ is the desired one.

5.14 Proposition [Strong $\beta T T$-normalization]: If $S$ is well-kinded in $\Gamma$, then there is no infinite $\beta T T$-reduction from $S$.

Proof: Assume, for a contradiction, that $R$ is an infinite $\beta T T$-reduction beginning from $S$. Let $R_0 = R$. Now repeat the following process as long as possible to construct a sequence $R_1, R_2, \ldots$ of infinite $\beta T$-reductions, all starting from $S$:

If $R_i$ contains no $\Gamma$-reduction that is immediately preceded by a $\beta T$-reduction, then stop. Otherwise, form $R_{i+1}$ from $R_i$ by using Lemma 5.12 repeatedly to move the first such $\Gamma$-reduction before any $\beta T$-reduction.

Note that all of the $R_i$ are infinite and that the first $i$ steps in each $R_i$ are all $\Gamma$-reductions. Now, there are two possibilities:

- The sequence of $R_i$'s eventually terminates, having reached some $R_n$ in which all $\Gamma$-reductions precede the first $\beta T$-reduction. But this means that $R_n$ contains only $\Gamma$-reductions, contradicting Lemma 5.4, or has an infinite tail consisting only of $\beta T$-reductions, contradicting Lemma 5.2.
- The sequence of $R_i$'s is infinite. But since each $R_i$ begins with at least $i$ $\Gamma$-reductions, we can use this to exhibit an infinite $\Gamma$-reduction beginning from $S$, contradicting Lemma 5.4.

\[\Box\]
6 Subtyping

Usually, in proof-theoretic analyses of calculi with subtyping, the subtyping relation itself presents the most challenging problems. This is also the case in $F^\subseteq$.

6.1 Proof Outline

Although the details of our development will be somewhat more involved, it is helpful to start by reviewing the standard argument [CG92, Ghe90, CMMS91, BCGS91, etc.] for the decidability of subtyping in the second-order system $F^\subseteq$:

1. Begin with an “original” presentation of the subtyping system that directly expresses its intended meaning, but which is not directly implementable.

2. Propose an alternative presentation of the same relation by a syntax-directed set of inference rules, in which the premises of each rule contain only metavariables whose values are uniquely determined by the form of the conclusion, and in which all the derivations of any given subtyping statement $\Gamma \vdash S \leq T$ must end with the same rule. (More precisely: more than one rule may be used to derive a given statement, as long as only one of them has premises whose applicability cannot be checked directly, without making any recursive calls.) This system can be implemented by a proof-search algorithm that will never have to guess or backtrack.

3. Check that this algorithm is indeed a decision procedure for the syntax-directed system by showing that proof search must terminate in finite time when started with any statement as its initial goal.

4. Show that the syntax-directed system is sound, in the sense that any subtyping statement derived by the algorithm is also derivable in the original system. This step is typically straightforward.

5. Finally, prove that the syntax-directed system is complete: that any statement derivable in the original system is also derivable by the algorithm. This step is where a deeper understanding is required.

The syntax-directed system may be viewed as a version of the original from which all “problematic” rules have been removed. In the case of $F^\subseteq$, there is just one such rule:

$$\frac{\Gamma \vdash S \leq U \quad \Gamma \vdash U \leq T} {\Gamma \vdash S \leq T} \quad \text{(S-Trans)}$$

By analogy with proof theory, this rule is sometimes called the cut rule of the subtyping system: the type $U'$ appearing in the subderivations is cut out when moving to the conclusion. By analogy with the sequent calculus or the simply typed $\lambda$-calculus (c.f. [GLT89]), this cut rule can be almost completely eliminated by rewriting derivations.

But not completely. In one situation, transitivity is actually essential. Statements with variables on the left-hand side cannot, in general, be proved without using transitivity. For example, $C \leq Top(\ast), B \leq C, A \leq B \vdash A \leq C$ must be proved using two instances of S-TVAR to establish the connections between $A$ and $B$ and between $B$ and $C$, which are then joined by a single instance of transitivity. Thus, to eliminate S-TRANS while retaining completeness, it is necessary to refine the treatment of variables, extending each instance of S-TVAR with an “internal” use of transitivity:

$$\frac{\Gamma \vdash \Gamma(A) \leq T} {\Gamma \vdash A \leq T} \quad \text{(S-TVAR-Plus-TRANS)}$$

It is easy to see that replacing S-TVAR with S-TVAR-Plus-TRANS in the original system does not affect its power. Moreover, S-TRANS can now be completely eliminated without losing any derivable statements. The resulting subtyping algorithm (i.e., the recursive procedure obtained from the syntax-directed system by ordering overlapping rules so that the “easy” ones come first) is:
The behavior of this algorithm reveals a great deal about the structure of the \( F_\leq \) subtyping relation. The first cases deal with the easy rules for \( \text{Top} \) and reflexivity. The third case says that a statement of the form \( \Gamma \vdash A \leq T \), where \( A \) is not identical to \( T \) and \( T \) is not \( \text{Top} \), can only be true if \( A \)'s upper bound is less than \( T \). In other words, the region between \( A \) and its upper bound is empty: there are no types strictly greater than \( A \) and strictly less than \( \Gamma(A) \).

Since this concept of "the smallest proper supertype of \( A \)" will also be crucial for our development, it is worth introducing some special notation for it. Write \( A \vdash \Gamma(A) \) for \( A \) promotes to \( \Gamma(S) \). We can then reformulate the enriched variable subtyping rule \( \text{S-TVar-Plus-Trans} \) as

\[
\frac{A \vdash \Gamma(A) \quad \Gamma \vdash S \leq T}{\Gamma \vdash A \leq T} \quad \text{(S-Promote-TVar)}
\]

or, more generally, as

\[
\frac{S \vdash U \quad \Gamma \vdash U \leq T}{\Gamma \vdash S \leq T} \quad \text{(S-Promote)}
\]

where the partial function \( \vdash \Gamma \) is undefined except on variables. The subtyping algorithm becomes:

\[
\text{check}(\Gamma \vdash S \leq T) =
\]

\[
\text{else if } S \vdash U \quad \text{then check}(\Gamma \vdash U \leq T)
\]

\[
\ldots
\]

Now, let us generalize these intuitions to the case of full \( F_\leq^w \). Here, we encounter one new kind of situation in which transitivity plays an essential role. For example, in the context

\[
\Gamma \equiv A \leq \text{Top}(\star), \; F \leq (\text{Fun}(B : \star) B),
\]

the statement \( \Gamma \vdash F \; A \leq A \) is provable as follows (ignoring kinding):

\[
\frac{\Gamma \vdash F \leq (\text{Fun}(B : \star) B)}{\text{S-TVar}} \quad \frac{\Gamma \vdash F \; A \leq (\text{Fun}(B : \star) B) \; A}{\text{S-App}} \quad \frac{\Gamma \vdash (\text{Fun}(B : \star) B) \; A \leq A}{\text{S-Conv}} \quad \frac{\Gamma \vdash F \; A \leq A}{\text{S-Trans}}
\]

The instance of transitivity in this derivation is again essential, but it is not an instance of the schema that motivated \( \text{S-TVar-Plus-Trans} \). In fact, it is possible to construct more involved examples where the instance of \( \text{S-TVar} \) is separated from the instance of \( \text{S-Trans} \) by arbitrarily many applications of \( \text{S-App} \). This suggests the following generalization of the promotion relation:

\subsection{6.1.1 Definition [Promotion]} The \emph{promotion} of a type \( A \; S_1 \ldots S_n \) in a well-formed context \( \Gamma \) is

\[\Gamma(A) \; S_1 \ldots S_n, \text{ written } A \; S_1 \ldots S_n \vdash \Gamma(A) \; S_1 \ldots S_n.\]
With this relation and S-Promote, both examples above can be derived without explicitly using S-Trans.

To extend the algorithm check to full $F_\omega$, one thing obviously missing is a clause for type abstraction matching the pointwise subtyping rule S-Abs. We add one as follows:

\[
\text{check}(\Gamma \vdash S \leq T) = \\
\ldots \\
\text{else if } S \equiv \text{Fun}(A; K_1) S_2 \text{ and } T \equiv \text{Fun}(A; K_1) T_2 \\
\text{then check}(\Gamma, A \leq \text{Top}(K_1) \vdash S_2 \leq T_2)
\]

Surprisingly, we do not need a similar clause for application,

\[
\text{check}(\Gamma \vdash S \leq T) = \\
\ldots \\
\text{else if } S \equiv S_1 \ T \text{ and } T \equiv T_1 \ U \\
\text{then check}(\Gamma \vdash S_1 \leq T_1)
\]

because its effect turns out to be covered by the promotion clause. But we do need to deal with the possibility of conversion; otherwise, for example, the statement $\Gamma \vdash (\text{Fun}(B; x) B) S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2$ will not be derivable.

Clearly, we need to perform some reduction on the arguments before choosing which clause of the algorithm to apply. We can make life easy by simply normalizing the arguments before looking at them.\(^5\)

Since the arguments to recursive calls in all of the clauses except promotion will remain in normal form if the original arguments are given in normal form, we only need to re-normalize in the promotion clause to preserve normality. The final algorithm, then, is:

\[
\text{check}(\Gamma \vdash S \leq T) = \\
\text{check}(\Gamma \vdash S' \leq T')
\]

\[
\text{check}(\Gamma \vdash S \leq T) = \\
\text{if } T \equiv \text{Top}(\text{kind}(S)) \\
\text{then true} \\
\text{else if } S \equiv T \\
\text{then true} \\
\text{else if } S \equiv \Gamma \ U \\
\text{then check}(\Gamma \vdash U' \leq T) \\
\text{else if } S \equiv S_1 \rightarrow S_2 \text{ and } T \equiv T_1 \rightarrow T_2 \\
\text{then check}(\Gamma \vdash T_1 \leq S_1) \\
\text{and check}(\Gamma \vdash S_2 \leq T_2) \\
\text{else if } S \equiv \text{All}(A \leq U) S_2 \text{ and } T \equiv \text{All}(A \leq U) T_2 \\
\text{then check}(\Gamma, A \leq U \vdash S_2 \leq T_2) \\
\text{else if } S \equiv \text{Fun}(A; K_1) S_2 \text{ and } T \equiv \text{Fun}(A; K_1) T_2 \\
\text{then check}(\Gamma, A \leq \text{Top}(K_1) \vdash S_2 \leq T_2) \\
\text{else false}.
\]

Our task for the remainder of the section will be to show that this algorithm is sound and complete for the rules in Section 2.4. Our first step is a technical reformulation of the original system, which provides a convenient setting for the arguments to follow: we remove the general rule of conversion and regain its effect

\(^5\)In a real implementation it is not desirable to fully normalize type expressions: this wastes time (in the vast majority of calls to the subtyping algorithm, the types being compared are identical) and results in the unnecessary expansion of type abbreviations, making the compiler's diagnostic output difficult for the programmer to understand. In practice, we reduce types only to weak head normal form, exposing only their outermost constructors at each step. The completeness of this modification rests on the observation that the reflexivity check in the algorithm can be restricted to type variables, applications, and the left-hand sides of quantifiers.

20
by generalizing each of the remaining rules to allow arbitrary reduction in the premises. For example, the
rule for subtyping arrow types
\[
\frac{\Gamma \vdash T_1 \leq S_1 \quad \Gamma \vdash S_2 \leq T_2}{\Gamma \vdash S_1 - S_2 \leq T_1 - T_2}
\]
becomes:
\[
\begin{array}{c}
S \xrightarrow{\beta T} S_1 - S_2 \\
\Gamma \vdash T_1 \leq S_1 \\
\Gamma \vdash S_2 \leq T_2
\end{array}
\quad
\begin{array}{c}
T \xrightarrow{\beta T} T_1 - T_2 \\
\Gamma \vdash S_1 \leq T_1 \\
\Gamma \vdash S_2 \leq T_2
\end{array}
\quad
\frac{\Gamma \vdash S \leq T}{\text{}}
\]
It is not hard to show that this reducing system is equivalent to the original. Next, we introduce two
important properties of certain derivations in the reducing system:

- A cut-free derivation is one with no instances of the rule of transitivity.
- A strong derivation is one in which every \( \xrightarrow{\beta T} \) reduction is actually a reduction to normal form.

Cut-free derivations are close to the form of those discovered by the algorithm check. Strength is a more
technical property, which reduces the complexity of the case analyses required at some crucial points. Using
these properties, the main facts that we need are:

1. The rule of transitivity can be eliminated from strong derivations.
2. The validity of subtyping is preserved by \( \beta T \)-reduction in cut-free derivations.

From these, we can show the central theorem: any derivation in the reducing system can be transformed to a
strong, cut-free derivation of the same statement. Finally, to obtain the algorithm, we observe that the
rule of application can also be eliminated from strong, cut-free derivations.

6.2 The Reducing System

The main difference between the reducing system and the original subtyping system presented in Section 2 is
that we remove the rule S-Conv and distribute its effects over the remaining rules in form of extra premises.
We also replace the rule S-TVar by the more general rule of promotion, R-Promote.

6.2.1 Definition [Reducing System]:

\[
\frac{S \xrightarrow{\beta T} U \quad T \xrightarrow{\beta T} U}{\Gamma \vdash S \leq T} \quad \text{(R-Refl)}
\]

\[
\frac{\Gamma \vdash S \leq U \quad \Gamma \vdash U \leq T \quad \Gamma \vdash U \notin K}{\Gamma \vdash U \leq T} \quad \text{(R-Trans)}
\]

\[
\frac{S \xrightarrow{\beta T} U \quad U \Rightarrow U' \quad \Gamma \vdash U' \leq T}{\Gamma \vdash S \leq T} \quad \text{(R-Promote)}
\]

\[
\frac{\Gamma \vdash S \leq T \quad \Gamma \vdash S \in K}{\Gamma \vdash S \leq T} \quad \text{(R-Top)}
\]

\[
\frac{S \xrightarrow{\beta T} S_1 \rightarrow S_2 \quad T \xrightarrow{\beta T} T_1 \rightarrow T_2}{\Gamma \vdash T_1 \leq S_1 \quad \Gamma \vdash S_2 \leq T_2} \quad \text{(R-Arrow)}
\]

\[
\frac{S \xrightarrow{\beta T} \text{All}(A \leq U) S_2 \quad T \xrightarrow{\beta T} \text{All}(A \leq U) T_2 \quad \Gamma, A \leq U \vdash S_2 \leq T_2}{\Gamma \vdash S \leq T} \quad \text{(R-All)}
\]

\[
\frac{S \xrightarrow{\beta T} \text{Fun}(A:K) S' \quad T \xrightarrow{\beta T} \text{Fun}(A:K) T' \quad \Gamma, A \leq \text{Top}(K) \vdash S' \leq T'}{\Gamma \vdash S \leq T} \quad \text{(R-Abs)}
\]

\[
\frac{S \xrightarrow{\beta T} S' U \quad T \xrightarrow{\beta T} T' U \quad \Gamma \vdash S' \leq T'}{\Gamma \vdash S \leq T} \quad \text{(R-App)}
\]
6.2.2 Notation: To avoid confusion, we sometimes distinguish derivations in different systems by marking the turnstile symbol $\vdash_\sigma$ for the original system, $\vdash_R$ for the reducing system, $\vdash_S$ for strong derivations in the reducing system, $\vdash_C$ for cut-free derivations in the reducing system, and $\vdash_{CS}$ for strong, cut-free derivations in the reducing system.

Our task for the remainder of Section 6.2 is to establish the equivalence of the reducing system and the original one. We begin by establishing some technical properties of the original.

6.2.3 Lemma [Kind invariance under promotion]: If $\Gamma \vdash S \in K$ and $\Gamma \vdash S' \in K$, then $\Gamma \vdash_\sigma S' \in K$.

6.2.4 Lemma [Promotion and Subtyping]: If $S \vdash S'$, then $\Gamma \vdash_\sigma S \leq S'$.

6.2.5 Lemma [Well-kind subderivations]:

1. Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $d$ is a derivation of $\Gamma \vdash_\sigma S \leq T$ and $d'$ is a derivation of $\Gamma' \vdash_\sigma S' \leq T'$, with $d'$ a subderivation of $d$, then $\Gamma' \vdash S', T' \in K$ for some $K$.

2. Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $d$ is a derivation of $\Gamma \vdash_R S \leq T$ and $d'$ is a derivation of $\Gamma' \vdash_R S' \leq T'$, with $d'$ a subderivation of $d$, then $\Gamma' \vdash S', T' \in K$ for some $K$.

6.2.6 Lemma: The following rules are derivable:

\[
\begin{align*}
\Gamma \vdash S, T & \in K \quad \Gamma \vdash_\sigma U \leq T \\
\Gamma \vdash U & \in K' \quad U =_\beta \tau S
\end{align*}
\]

\[
\Gamma \vdash S \leq T
\]

\[
\begin{align*}
\Gamma \vdash S' & \in K' \quad S =_\beta \tau S' \quad T =_\beta \tau T'
\end{align*}
\]

\[
\Gamma \vdash_\sigma S \leq T
\]

\[
\begin{align*}
\Gamma \vdash U & \leq T \\
\Gamma \vdash U & \in K'
\end{align*}
\]

\[
\Gamma \vdash S \leq T
\]

(S-Conv(1)) \[ S-Conv(2) \]

Proof: For S-Conv (1):

\[
\begin{align*}
\Gamma \vdash_\sigma S & \leq S \\
\Gamma \vdash S & \in K \\
S & =_\beta \tau U
\end{align*}
\]

\[
\Gamma \vdash_\sigma S \leq U
\]

\[
\begin{align*}
\Gamma \vdash U & \leq T \\
\Gamma \vdash U & \in K'
\end{align*}
\]

\[
\Gamma \vdash_\sigma S \leq T
\]

S-Conv S-Conv S-Trans

For S-Conv (2), the proof is similar. \[ \Box \]

6.2.7 Lemma [Equivalence]: The original and the intermediate subtyping systems are equivalent for well-kindred types: if $\Gamma \vdash S, T \in K$, then $\Gamma \vdash_\sigma S \leq T$ iff $\Gamma \vdash_R S \leq T$.

Proof: By induction on derivations. \[ \Box \]

For the remainder of Section 6.2, we will work exclusively within the reducing system.

6.3 Cut Elimination

We begin our analysis of the reducing system with a proof that R-Trans is inessential: any derivation ending with it can be rewritten as a derivation in the cut-free subsystem. To control the complexity of the combinatorial analysis, we do not show this property for arbitrary uses of transitivity, but only for uses of a restricted form: we consider only cut-terms in normal form and ask that the subderivations of the cut be strong. The following section will show that these conditions can always be achieved.

With these restrictions, the proof of cut elimination is a straightforward extension of standard proofs for the second-order fragment (c.f. [BCGS91, CG92, CMMS91, CP94]). We begin with one technical lemma.
6.3.1 Lemma: Suppose \( \Gamma \vdash S \in K_S \) and \( \Gamma \vdash A T_1 \ldots T_n \in K_T \). If \( \Gamma \vdash_{CS} S \leq A T_1 \ldots T_n \), then this statement can be proved using a sequence of instances of R-PROMOTE preceded by a single instance of R-REFL.

\[
S_n \rightarrow_{\beta_T} S \quad A T_1 \ldots T_n \rightarrow_{\beta_T} S
\]

Proof: By induction on the combined size of the given subderivations. The R-REFL case is immediate; R-PROMOTE makes straightforward use of the induction hypothesis. R-APP uses the induction hypothesis to construct a derivation of the required shape for the left-hand sides of the application \( (\Gamma \vdash_{CS} S' \leq A T'_1 \ldots T'_{n-1}) \); it is then easy to check that the right-hand side \( (T'_n) \) can be adjoined to all the steps in this derivation.

6.3.2 Proposition [Cut elimination]: Suppose \( \Gamma \vdash S \in K_S \) and \( \Gamma \vdash T \in K_T \) and \( \Gamma \vdash U \in K_U \). If \( \Gamma \vdash_{CS} S \leq U \) and \( \Gamma \vdash_{CS} U \leq T \), where \( U \) is in normal form, then \( \Gamma \vdash_{CS} S \leq T \).

Proof: By induction on the combined size of the given subderivations. Proceed by a case analysis on the last rule in each.

Case: R-REFL on the left / anything on the right:

\[
S \rightarrow_{\beta_T} U \quad U \rightarrow_{\beta_T} U
\]

\[\Gamma \vdash_{CS} S \leq U\]

To prove \( \Gamma \vdash_{CS} S \leq T \), we can directly use the subderivations of \( \Gamma \vdash_{CS} U \leq T \), since \( S \rightarrow_{\beta_T} U \).

Case: Anything on the left / R-REFL on the right:

Similar.

Case: Anything on the left / R-TOP on the right:

\[
T \rightarrow_{\beta_T} \text{Top}(K) \quad \Gamma \vdash_{CS} U \in K
\]

\[\Gamma \vdash_{CS} U \leq T\]

By the well-kindness of subderivations, we obtain \( \Gamma \vdash_{CS} S \leq T \) immediately by R-TOP.

Case: R-PROMOTE on the left / anything on the right:

\[
S \rightarrow_{\beta_T} \text{top} S' \quad \Gamma \vdash_{CS} S' \leq U
\]

\[\Gamma \vdash_{CS} S \leq U\]

By induction (using well-kindness of subderivations) and R-PROMOTE.

Case: Anything on the left / R-PROMOTE on the right:

\[
U = A U_1 \ldots U_n \quad \Gamma \vdash_{CS} V \leq T
\]

\[\Gamma \vdash_{CS} U \leq T\]

By Lemma 6.3.1, we may assume that the derivation of \( \Gamma \vdash_{CS} S \leq U \) consists of a sequence of instances of R-PROMOTE preceded by an instance of R-REFL:

\[
S \rightarrow_{\beta_T} S' \quad U = A U_1 \ldots U_n
\]

\[\Gamma \vdash_{CS} S_n \leq A U_1 \ldots U_n\]

\[
\vdots
\]

\[
S \rightarrow_{\beta_T} S' \quad \Gamma \vdash_{CS} S_1 \leq A U_1 \ldots U_n
\]

\[\Gamma \vdash_{CS} S \leq A U_1 \ldots U_n\]
Replacing the final instance of R-Refl by an additional instance of R-Promote, we obtain

\[
\begin{align*}
S_n \xrightarrow{\beta T} & V \quad \Gamma \vdash_{CS} V \leq T \\
S_{n-1} \xrightarrow{\beta T} S_{n-1} & \vdash S_n \quad \Gamma \vdash_{CS} S_n \leq T \\
& \vdots \\
S \xrightarrow{\beta T} S' & \vdash S_1 \quad \Gamma \vdash_{CS} S_1 \leq T \\
& \vdash_{CS} S \leq T
\end{align*}
\]

as desired.

This takes care of 29 of the 49 cases. The remaining cases are listed in the following table.

<table>
<thead>
<tr>
<th></th>
<th>R-Arrow</th>
<th>R-All</th>
<th>R-Abs</th>
<th>R-App</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-TOP</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>R-ARROW</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>R-ALL</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>R-ABS</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>R-APP</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
</tr>
</tbody>
</table>

The cases marked √ are dealt with individually below. Those marked × can never occur, since the two rules in question would place incompatible constraints on the form of \( U \). (This can easily be checked by inspecting the rules; the assumption that the given derivations are strong is needed in several places.)

**Case R-ALL:**

\[
\begin{align*}
S & \xrightarrow{\beta T} \text{All}(A \leq U_1) S_2 \quad U = \text{All}(A \leq U_1) U_2 \\
\Gamma, A \leq U_1 & \vdash S_2 \leq U_2 \\
\Gamma \vdash_{CS} S \leq U \\
U = \text{All}(A \leq U_1) U_2 & \quad T \xrightarrow{\beta T} \text{All}(A \leq U_1) T_2 \\
\Gamma, A \leq U_1 & \vdash U_2 \leq T_2 \\
\Gamma \vdash_{CS} U \leq T
\end{align*}
\]

By the well-kindness of subderivations, the induction hypothesis applies, giving:

\[
\begin{align*}
S \xrightarrow{\beta T} \text{All}(A \leq U_1) S_2 \\
\Gamma, A \leq U_1 & \vdash S_2 \leq T_2 \\
\Gamma \vdash_{CS} S \leq T
\end{align*}
\]

**Case R-ARROW, R-ABS:**

Similar.

**Case R-APP:**

\[
\begin{align*}
S & \xrightarrow{\beta T} S' V \quad U = U' V \\
\Gamma \vdash_{CS} S' \leq U' \\
\Gamma \vdash_{CS} U \leq T' \\
U & = U' V \quad T \xrightarrow{\beta T} T' V \\
\Gamma \vdash_{CS} U \leq T' \\
\Gamma \vdash_{CS} T \vdash U \leq T
\end{align*}
\]

By the well-kindness of subderivations, the induction hypothesis applies, giving:

\[
\begin{align*}
S \xrightarrow{\beta T} S' V \\
T \xrightarrow{\beta T} T' V \\
\Gamma \vdash_{CS} S' \leq T' \\
\Gamma \vdash_{CS} S \leq T
\end{align*}
\]
6.4 Reduction and Subtyping

The main task of this subsection is to show that, for cut-free derivations, $\beta T$-reduction in types does not interfere with the subtyping judgement.

The cornerstone of the argument is a substitution lemma saying (informally) that if $S \leq T$ then $[V/A]S \leq [V/A]T$. From this, we can show that the reduction of an outermost redex on either the left-hand or the right-hand side of a subtyping statement preserves its derivability. As in the proof of Church-Rosser in Section 3, we extend these properties to a proof of the preservation of subtyping under arbitrary multi-step reduction by passing through an intermediate step where we show it for one-step parallel reduction.

In outline, then, the major steps are as follows:

- Reduction (6.4.9)
- Parallel reduction (6.4.8)
- Outer reduction (left) (6.4.7)
- Outer reduction (right) (6.4.5)
- Substitution (6.4.3)

We begin with two technical lemmas:

6.4.1 Lemma [Expansion preserves subtyping]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash \text{Top}(K) S_1 \ldots S_n \leq S$ and $\Gamma \vdash \text{Top}(K) T_1 \ldots T_n \leq T$, then $\Gamma \vdash \text{Top}(K) S_1 \ldots S_n \leq T$.

Proof: By inspection of the rules, using the properties of kinding and reduction. \hfill \Box

6.4.2 Lemma [Maximality of Top]: Suppose $\Gamma \vdash \text{Top}(K) S_1 \ldots S_n \leq S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash \text{Top}(K) S_1 \ldots S_n \leq T$, then $T \rightarrow^* \text{Top}(K)$. 

Proof: Straightforward induction, using the properties of kinding and reduction. \hfill \Box

Now we come to the key result of this subsection: the preservation of subtyping under substitution.

6.4.3 Lemma [Substitution preserves subtyping]: Let $\Gamma = \Gamma_1, A \leq \text{Top}(K), \Gamma_2$ and $\Gamma' = \Gamma_1, [V/A] \Gamma_2$. Suppose that $\Gamma \vdash V \in K$ and that $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash \text{Top}(K) S \leq T$, then $\Gamma' \vdash [V/A]S \leq [V/A]T$.

Proof: The cases other than R-PROMOTE are straightforward; we give the argument for R-ALL as an example.\(^6\)

Case R-ALL:

$$S \rightarrow^*_{\beta T} \text{All}(A' \leq U) S_2 \quad T \rightarrow^*_{\beta T} \text{All}(A' \leq U) T_2$$

$$\Gamma, A' \leq U \vdash_c S_2 \leq T_2$$

$$\Gamma \vdash \text{C} \equiv S \leq T$$

By Lemma 3.3(3), we have

$$[V/A]S \rightarrow^*_{\beta T} \text{All}(A' \leq [V/A]U) [V/A]S_2$$
$$[V/A]T \rightarrow^*_{\beta T} \text{All}(A' \leq [V/A]U) [V/A]T_2$$

By the well-kindled subderivations lemma and the induction hypothesis, $\Gamma', A' \leq [V/A]U \vdash_c [V/A]S_2 \leq [V/A]T_2$. Now R-ALL applies as follows:

$$[V/A]S \rightarrow^*_{\beta T} \text{All}(A' \leq [V/A]U) [V/A]S_2$$
$$[V/A]T \rightarrow^*_{\beta T} \text{All}(A' \leq [V/A]U) [V/A]T_2$$

$$\Gamma', A' \leq [V/A]U \vdash_c [V/A]S_2 \leq [V/A]T_2$$

$$\Gamma \vdash \text{C} \equiv [V/A]S \leq [V/A]T$$

\(^6\)Note that the more general form of this property, in which $\Gamma(A)$ is allowed to be any supertype of $V$, would be much more difficult to prove. Here we obtain a straightforward proof by considering only the form that will actually be required later: in the critical case — the one for R-PROMOTE — the fact that $\Gamma(A) = \text{Top}(K)$ allows a direct argument using Lemma 6.4.2.
Case R-Promote:

\[ S \rightarrow_{\beta_T}^{\ast} S' \quad S' \vdash_T U \quad \Gamma \vdash C U \leq T \]

\[ \Gamma \vdash C S \leq T \]

The definition of promotion gives \( S' = A' U_1 \ldots U_n \) for some \( n \geq 0 \), and \( U = \Gamma(A') U_1 \ldots U_n \). Now there are two subcases to consider:

Subcase: \( A' \neq A \)

By induction, \( \Gamma' \vdash_C [V/A](\Gamma(A') U_1 \ldots U_n) \leq [V/A]T \). Now,

\[ [V/A](\Gamma(A') U_1 \ldots U_n) = ([V/A](\Gamma(A'))) [V/A](U_1 \ldots [V/A]U_n) = \Gamma'(A') [V/A]U_1 \ldots [V/A]U_n, \]

and by Lemma 3.3(3), \( [V/A]S \rightarrow_{\beta_T}^{\ast} A' [V/A]U_1 \ldots [V/A]U_n \). So, by the definition of promotion, we can apply R-Promote to obtain the desired result as follows:

\[ [V/A]S \rightarrow_{\beta_T}^{\ast} A' [V/A]U_1 \ldots [V/A]U_n \]

\[ \Gamma' \vdash_C [V/A](\Gamma(A') U_1 \ldots U_n) \leq [V/A]T \]

\[ \Gamma' \vdash_C [V/A]S \leq [V/A]T \]

Subcase: \( A' = A \)

Since \( \Gamma(A) = \text{Top}(K) \), we have \( \Gamma \vdash_C \text{Top}(K) U_1 \ldots U_n \leq T \). By the well-kinded subderivations lemma (6.2.5), \( U \) is well-kinded in \( \Gamma \). So by Lemma 6.4.2, we have \( T \rightarrow_{\beta_T}^{\ast} \text{Top}(K_T) \). By Lemma 3.3, also \( [V/A]T \rightarrow_{\beta_T}^{\ast} \text{Top}(K_T) \). By the well-kindedness of subderivations, \( \Gamma \vdash S \in K_T \), so by the fact that substitution preserves kinding (Lemma 4.11), rule R-Top

\[ [T/A]T \rightarrow_{\beta_T}^{\ast} \text{Top}(K_T) \quad \Gamma \vdash [V/A]S \in K_T \]

\[ \Gamma' \vdash_C [V/A]S \leq [V/A]T \]

This gives us the desired result. \( \square \)

The next lemma introduces a technical property needed for the following one: a subtype of a Fun-type is either a Fun-type itself or can be promoted to one in a finite number of steps.

6.4.4 Lemma [Fun-right]: Suppose \( \Gamma \vdash S \in K_S \) and \( \Gamma \vdash T \in K_T \). If \( \Gamma \vdash_C S \leq \text{Fun}(A;K)T \), then this statement can be derived by a sequence of instances of R-PROMOTE preceded by one instance of R-FUN:

\[ S \rightarrow_{\beta_T}^{\ast} \text{Fun}(A;K)S \]

\[ \Gamma \vdash_C S \leq \text{Fun}(A;K)T \]

Proof: By induction on a derivation of \( \Gamma \vdash_C S \leq \text{Fun}(A;K)T \). By the form of the right-hand side and the fact that the derivation is cut free, there are three cases to consider. (Note, in passing, that this is one point where we crucially depend on the absence of cut: this argument fails on arbitrary reducing derivations.)

Case R-RefL:

\[ S \rightarrow_{\beta_T}^{\ast} U \quad \text{Fun}(A;K)T \rightarrow_{\beta_T}^{\ast} U \]

\[ \Gamma \vdash C S \leq \text{Fun}(A;K)T \]

By definition of \( \rightarrow_{\beta_T}^{\ast} \), the type \( U \) must be of the form \( \text{Fun}(A;K)T' \), with \( T \rightarrow_{\beta_T}^{\ast} T' \). So, by R-Abs and R-RefL:

\[ S \rightarrow_{\beta_T}^{\ast} \text{Fun}(A;K)T \quad \text{Fun}(A;K)T \rightarrow_{\beta_T}^{\ast} \text{Fun}(A;K)T' \]

\[ \Gamma, A \leq \text{Top}(K) \vdash_C T \leq T' \]

\[ \Gamma \vdash C S \leq \text{Fun}(A;K)T \]
Case R-Abs:
Immediate.

Case R-Promote:

\[
S \rightarrow^{\beta_T} S' \mid \Gamma \quad \Gamma \vdash C \quad S \leq \text{Fun}(A:K) T
\]
\[
\Gamma \vdash C S \leq \text{Fun}(A:K) T
\]

By the well-kindness of subderivations lemma, the induction hypothesis, and R-Promote.

\[\square\]

6.4.5 Lemma [Outer reduction on the right]: Suppose \(\Gamma \vdash S \in K_S\) and \(\Gamma \vdash \text{(Fun}(A:K) T) U \in K_T\). If \(\Gamma \vdash C S \leq \text{(Fun}(A:K) T) U\), then \(\Gamma \vdash C S \leq [U/A] T\).

Proof: By induction on a proof of \(\Gamma \vdash C S \leq \text{(Fun}(A:K) T) U\). The only difficult case is R-App, which we give in full below. R-Refl uses the Church-Rosser property. R-Promote uses only the induction hypothesis. All the other cases follow a common pattern: we show just R-All.

Case R-All:

\[
S \rightarrow^{\beta_T} \text{All}(A' \leq V) S_2 \quad \text{(Fun}(A:K) T) U \rightarrow^{\beta_T} \text{All}(A' \leq V) T_2
\]
\[
\Gamma, A' \leq V \vdash C S_2 \leq T_2
\]
\[
\Gamma \vdash C S \leq \text{(Fun}(A:K) T) U
\]

By the outermost reduction corollary (3.5), \([U/A] T \rightarrow^{\beta_T} \text{All}(A' \leq V) T_2\). The result follows immediately by R-All.

Case R-App:

There are two cases to consider, depending on whether the redex \((\text{Fun}(A:K) T) U\) is itself contracted at some point in the reduction sequence \([\text{Fun}(A:K) T) U \rightarrow^{\beta_T} \cdots\).

Subcase: \((\text{Fun}(A:K) T) U \rightarrow^{\beta_T} \text{Fun}(A:K) T) U\)
\[
\Gamma \vdash C S \leq \text{(Fun}(A:K) T) U
\]
\[
\Gamma \vdash C W \leq \text{Fun}(A:K) T
\]

Since \([U/A] T \rightarrow^{\beta_T} \text{Fun}(A:K) T) U\), the result follows immediately from R-App.

Subcase: \(S \rightarrow^{\beta_T} W U\)
\[
\Gamma \vdash C W \leq \text{Fun}(A:K) T
\]

Here we cannot directly use the subderivation of \(\Gamma \vdash C W \leq \text{Fun}(A:K) T\). Intuitively, we must “look inside” this derivation to find an inner subderivation in which \(T'\) appears by itself on the right-hand side, and use this to rebuild a subderivation ending with \([U/A] T\) on the right. (More precisely, the inner subderivation will have \(T''\) on the right, where \(T' \rightarrow^{\beta_T} T''\).) This we accomplish as follows.

First, we use the Fun-right lemma (6.4.4) to obtain a derivation of \(\Gamma \vdash C W \leq \text{Fun}(A:K) T\) in a very rigid form: a sequence of \(n\) instances of R-Promote ending with an instance of R-Abs:

\[
W_n \rightarrow^{\beta_T} \text{Fun}(A:K) X
\]
\[
\text{Fun}(A:K) T \rightarrow^{\beta_T} \text{Fun}(A:K) T''
\]
\[
\Gamma, A \leq \text{Top}(K) \vdash C X \leq T''
\]
\[
W_n \rightarrow^{\beta_T} \mid \Gamma W_n
\]
\[
\Gamma \vdash C W_n \leq \text{Fun}(A:K) T''
\]
\[
\vdots
\]
\[
W \rightarrow^{\beta_T} \mid \Gamma W_1
\]
\[
\Gamma \vdash C W_1 \leq \text{Fun}(A:K) T'
\]
\[
\Gamma \vdash C W \leq \text{Fun}(A:K) T'
\]

From the instance of R-Abs at the top, we obtain \(\Gamma \vdash C [U'/A] X \leq [U'/A] T''\) by the substitution lemma (6.4.3). (To check that the lemma applies, we need the following observations: (1) \(\Gamma \vdash U' \in K\) by the assumption \(\Gamma \vdash \text{Fun}(A:K) T) U \in K_T\), the fact that \(U \rightarrow^{\beta_T} U''\), the generation lemma for types, and subject reduction. (2) \(X\) and \(T''\) are both well-kindred in \(\Gamma, A \leq \text{Top}(K)\), by the well-kindness of subderivations.)

At this point, we have accomplished a substitution operation, but not quite the one we need. We must now work backwards to the desired result.
From $\Gamma \vdash_{\mathcal{C}} [U/A][X] \leq [U/A][T']$ and the fact that $[U/A][T] \vdash_{\beta} [U'/A][T']$ (Lemma 3.3), we use the expansion lemma (6.4.1) to obtain $\Gamma \vdash_{\mathcal{C}} [U'/A]/X \leq [U'/A][T]$, after remarking that $[U'/A]/X$ and $[U'/A][T]$ are well-kind by subject reduction. This gives us what we need on the right-hand side; now we turn to fixing the left-hand side.

$W_n U'$ is well-kind and $W_n U' \vdash_{\beta} ([U](A:K) X) U' \vdash_{\beta} [U'/A][T]$, so by the expansion lemma, $\Gamma \vdash_{\mathcal{C}} W_n U' \leq [U'/A][T]$. We use this as the starting point for a new sequence of R-PROMOTE steps where an application to $U'$ has been added to each intermediate term:

$$W_{n-1} U' \vdash_{\beta} [U/n] W_n U' \quad \Gamma \vdash_{\mathcal{C}} W_n U' \leq [U'/A][T]$$

The conclusion of this derivation is the desired statement.

We proceed for the left-hand side of $\leq$ in the same way as we have done for the right-hand side.

6.4.6 Lemma [Fun-left]: Suppose $\Gamma \vdash Fun(A:K)S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} Fun(A:K)S \leq T$, then one of the following cases holds:

1. $T \vdash_{\beta} \top_{\mathcal{C}}$ Top($K_T$); or

2. $Fun(A:K)S \vdash_{\beta} Fun(A:K)S'$ and $T \vdash_{\beta} Fun(A:K)T'$, with $\Gamma, A \leq \top_{\mathcal{C}} S' \leq T'$.

Proof: By inspection of the rules. By the form of the left-hand side of the statement and the fact that we consider only cut-free derivations, there are only three cases to consider:

Case R-Refl:

$$Fun(A:K)S \vdash_{\beta} U \quad T \vdash_{\beta} U$$

$U$ must be of the form $Fun(A:K)S'$ where $S \vdash_{\beta} S'$ so we can conclude:

$$S' \vdash_{\beta} S' \quad S' \vdash_{\beta} S'$$

$$\Gamma, A \leq \top_{\mathcal{C}} S' \leq S'$$

Case R-Top, R-Abs:

By case 1 and 2, respectively. □

6.4.7 Lemma [Outer reduction on the left]: Suppose $\Gamma \vdash (Fun(A:K)S) U \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} (Fun(A:K)S) U \leq T$, then $\Gamma \vdash_{\mathcal{C}} [U/A][S] \leq T$.

Proof: By induction. Again, R-App is the only difficult case. If the reduction starting from $(Fun(A:K)S) U$ reduces the outer redex at some point, then this case proceeds as in the analogous case on the right-hand side (6.4.5). Otherwise, we have:

$$(Fun(A:K)S) U \vdash_{\beta} (Fun(A:K)S') U' \quad T \vdash_{\beta} W U'$$

$$\Gamma \vdash_{\mathcal{C}} (Fun(A:K)S') U' \leq W$$

$$\Gamma \vdash_{\mathcal{C}} (Fun(A:K)S) U \leq T$$

By the well-kindness of subderivations, both $Fun(A:K)S'$ and $W$ are well kinded. By the Fun-left lemma, there are two cases to consider:

1. We are given $W' \vdash_{\beta} \top_{\mathcal{C}} (K \rightarrow K_T)$. Since $\Gamma \vdash [U/A][S] \in K_T$ (by subject reduction and uniqueness of kinding) and $T \vdash_{\beta} W V$, we can conclude $\Gamma \vdash_{\mathcal{C}} [U/A][S] \leq T$ by R-Top.
2. We are given

\[
\begin{align*}
\text{Fun}(A; K) & \to \beta S' \quad \text{Fun}(A; K) \to S'' \\
W & \to \beta \text{Fun}(A; K) Y \\
\Gamma, A \leq \text{Top}(K) & \vdash C \to S'' \leq Y.
\end{align*}
\]

Both \(S''\) and \(Y\) are well kinded in \(\Gamma, A \leq \text{Top}(K)\) (by well kindedness of subderivations, subject reduction, and the generation lemma), so \(\Gamma \vdash C \to U'' / A \to U' / A \to Y\) by the substitution lemma (6.4.3).

As in the analogous case on the right, we have now accomplished a substitution operation, but not quite the one we need. We must again work backwards to the desired result.

From \(\Gamma \vdash C \to U'' / A \to U' / A \to Y\) and the fact that \([U / A]S \to \beta T [U'' / A]S''\) (Lemma 3.3), we can use the expansion lemma to obtain \(\Gamma \vdash C \to U'' / A \to Y\). (Note that \([U / A]S\) is well kinded by subject reduction and \([U'' / A]Y\) by subject reduction and the generation lemma.) Then we have

\[
T \to \beta U \quad V \to \beta (\text{Fun}(A; K) Y) \to U'' \to \beta [U'' / A]Y,
\]

which yields the desired result by the expansion lemma. \(\Box\)

Using the two outer reduction lemmas, we can do one reduction step at the outside of a type while preserving the subtyping judgement. The next step is to generalize this to an arbitrary reduction. For this purpose, we choose the relation \(\to \beta T\).

6.4.8 Proposition [Parallel reduction preserves subtyping]: Suppose \(\Gamma \vdash S \in K_S\) and \(\Gamma \vdash T \in K_T\).

1. If \(\Gamma \vdash C \to U \to \beta T\) \(S\) and \(\Gamma \to \beta T\) \(\Gamma'\), then \(\Gamma' \vdash C \to S' \leq T\).

2. If \(\Gamma \vdash C \to U \to \beta T\) \(T'\) and \(\Gamma \to \beta T\) \(\Gamma'\), then \(\Gamma' \vdash C \to S \leq T'\).

(Parallel reduction of contexts, written \(\Gamma \to \beta T\) \(\Gamma'\), is the pointwise extension of parallel reduction of types.)

**Proof:** By simultaneous induction on derivations. The only difficult cases are R-App and R-Promote. We show only the arguments for part (1), where \(S \to \beta T\) \(S'\); part (2) is similar except in the case R-Promote, which is easier.

**Case R-Refl:**

\[
S \to \beta U \quad T \to \beta U
\]

By Lemma 3.8 there is a type \(U\) with \(U \to \beta U\) \(U'\) and \(S' \to \beta U'\) as well as \(T \to \beta U'.\) The result follows by R-Refl.

**Case R-Top:**

\[
T \to \beta \text{Top}(K) \quad \Gamma \vdash S \in K_S \quad \Gamma \vdash C \to S \leq T
\]

By subject reduction and uniqueness of kinding we have \(K' = K_T\). So by subject reduction, the well-kindness of subderivations, and Lemma 4.12, the result follows by R-Top.

**Case R-Arrow:**

\[
S \to \beta S_1 \quad S_1 \to S_2 \quad T \to \beta T_1 \quad T_1 \to T_2 \\
\Gamma \vdash C \to S \leq T
\]

By Lemma 3.8 and the definition of parallel reduction (3.1), there are types \(S_1\) and \(S_2\) with \(S_1 \to \beta S_1\) and \(S_2 \to \beta S_2\) and:

\[
\begin{align*}
S & \to \beta S_1 \\
S_1 & \to \beta S_1 \\
S_1 & \to S_1 - S_2 \\
S_1 & \to S_1 - S_2
\end{align*}
\]

Since the \(S_i\) and \(T_i\) are well-kindled, the result follows by induction.
Case R-All:

\[ S \xrightarrow{\beta_T} \text{All}(A \leq U) V \quad T \xrightarrow{\beta_T} \text{All}(A \leq U) W \]
\[ \Gamma, A \leq U \vdash T \leq W \]
\[ \Gamma \vdash_c S \leq T \]

By Lemma 3.8 and the definition of parallel reduction (3.1), there are types \( U' \) and \( V' \) with \( U \xrightarrow{\beta_T} U' \) and \( V \xrightarrow{\beta_T} V' \) and:

\[
\begin{array}{ccc}
S & \xrightarrow{\beta_T} & S' \\
\downarrow{\beta_T} & & \downarrow{\beta_T} \\
\text{All}(A \leq U) V & \xrightarrow{\beta_T} & \text{All}(A \leq U') V'
\end{array}
\]

Since the \( V \) and \( W \) are well-kinded in \( \Gamma, A \leq U \), the result follows by induction:

\[
\begin{array}{ccc}
S' & \xrightarrow{\beta_T} & \text{All}(A \leq U') V' \\
\downarrow{\beta_T} & & \downarrow{\beta_T} \\
T & \xrightarrow{\beta_T} & \text{All}(A \leq U') W
\end{array}
\]
\[ \Gamma', A \leq U' \vdash T' \leq W \]
\[ \Gamma' \vdash_c S' \leq T' \]

Case R-All, R-Abs:

Similar.

Case R-Promote:

\[
\begin{array}{cccc}
S & \xrightarrow{\beta_T} & W \\
\downarrow{\beta_T} & & \downarrow{\beta_T} \\
W & \vdash \Gamma \quad & U & \leq T
\end{array}
\]
\[ \Gamma \vdash_c S \leq T \]

By the definition of the promotion relation, we have \( W = A \ S_1 \ldots S_n \vdash \Gamma \ A \ S_1 \ldots S_n = U \). By Lemma 3.8 and the definition of \( \xrightarrow{\beta_T} \), there are \( S'_1 \ldots S'_n \) with \( S_i \xrightarrow{\beta_T} S'_i \) and:

\[
\begin{array}{ccc}
S & \xrightarrow{\beta_T} & S' \\
\downarrow{\beta_T} & & \downarrow{\beta_T} \\
A \ S_1 \ldots S_n & \xrightarrow{\beta_T} & A \ S'_1 \ldots S'_n
\end{array}
\]

Let \( U' = \Gamma'(A) \ S'_1 \ldots S'_n \). By well-kindedness of subderivations and subject reduction, \( U \) and \( U' \) are well-kinded. The result follows by induction, the fact that \( \Gamma(A) \ S_1 \ldots S_n \xrightarrow{\beta_T} \Gamma'(A) S'_1 \ldots S'_n \), and R-Promote.

Case R-App:

\[
\begin{array}{cccc}
S & \xrightarrow{\beta_T} & U \ W \\
\downarrow{\beta_T} & & \downarrow{\beta_T} \\
T & \xrightarrow{\beta_T} & V \ W \\
\downarrow{\beta_T} & & \downarrow{\beta_T} \\
\Gamma \vdash_c U \leq V
\end{array}
\]
\[ \Gamma \vdash_c S \leq T \]

By Lemma 3.8, there is some \( X \) such that:

\[
\begin{array}{ccc}
S & \xrightarrow{\beta_T} & S' \\
\downarrow{\beta_T} & & \downarrow{\beta_T} \\
U \ W & \xrightarrow{\beta_T} & X
\end{array}
\]

By subject reduction, \( \Gamma \vdash U \ W \in K_S \) and \( \Gamma \vdash V \ W \in K_T \). By the generation lemma, this implies that \( U \) and \( V \) are well-kinded. Continue by cases on the form of \( U \). (The interesting one is when \( U \) is a type operator.)
Subcase: $U = A$ or $U = U_1 \ U_2$
By the definition of parallel reduction, $X$ must have the form $U' \ W'$ with $U \xrightarrow{\beta_T} U'$ and $W \xrightarrow{\beta_T} W'$. By subject reduction, $U'$ is well kinded. The result then follows by induction and R-APP:

$$S' \xrightarrow{\beta_T} U' \ W' \quad T \xrightarrow{\beta_T} V \ W \quad \Gamma \vdash_c U' \leq V \quad \Gamma \vdash_c S' \leq T$$

Subcase: $U = \text{Top}(K')$

$$S \xrightarrow{\beta_T} \text{Top}(K') \ W \quad T \xrightarrow{\beta_T} V \ W \quad \Gamma \vdash_c \text{Top}(K') \leq V$$

By Lemma 6.4.2, $V \xrightarrow{\beta_T} \text{Top}(K')$. Since $T$ has kind $K_T$, so does $\text{Top}(K') \ W$, and, by Lemma 4.10, $T \xrightarrow{\beta_T} \text{Top}(K') \ W \xrightarrow{\beta_T} \text{Top}(K_T)$. The result then follows by R-Top (using subject reduction, well-kindness of subderivations, and Lemma 4.12):

$$T \xrightarrow{\beta_T} \text{Top}(K_T) \quad \Gamma \vdash S' \in K_T \quad \Gamma' \vdash_c S' \leq T$$

Subcase: $U = \text{Fun}(A;K) \ U_1$

We have $(\text{Fun}(A;K) \ U_1) \ W \xrightarrow{\beta_T} X$. By the definition of parallel reduction, $X$ can have one of two forms:

Subsubcase: $X = (\text{Fun}(A;K) \ U_1) \ W'$

The result follows by induction (using subject reduction and generation for the kinded hypothesis) and R-APP:

$$S' \xrightarrow{\beta_T} (\text{Fun}(A;K) \ U_1) \ W' \quad T \xrightarrow{\beta_T} V \ W \xrightarrow{\beta_T} V' \ W' \quad \Gamma' \vdash_c \text{Fun}(A;K) \ U_1 \leq V \quad \Gamma' \vdash_c S' \leq T$$

Subsubcase: $X = [W'/A] \ U_1$

$$U_1 \xrightarrow{\beta_T} U'_1 \quad W \xrightarrow{\beta_T} W'$$

Then

$$\Gamma \vdash_c \text{Fun}(A;K) \ U_1 \leq V \quad \Rightarrow \quad (\text{by induction})$$
$$\Gamma' \vdash_c \text{Fun}(A;K) \ U_1' \leq V \quad \Rightarrow \quad (\text{by R-App})$$
$$\Gamma' \vdash_c \text{Fun}(A;K) \ U_1' \ W' \leq V \ W' \quad \Rightarrow \quad (\text{by outer reduction on the left (6.4.7)})$$
$$\Gamma' \vdash_c [W'/A] \ U_1' \leq V \ W' \quad \Rightarrow \quad (\text{by the expansion lemma (6.4.1)})$$
$$\Gamma' \vdash_c S' \leq T,$$}

using subject reduction and the generation lemma for the required kinded hypotheses in the first step and subject reduction in the third and fourth steps. \hfill \Box

6.4.9 Corollary [Reduction preserves subtyping]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $S \xrightarrow{\beta_T} S'$ and $T \xrightarrow{\beta_T} T'$, with $\Gamma \vdash_c S \leq T$, then $\Gamma \vdash_c S' \leq T'$.

\footnote{Here we can be more precise about why we choose parallel reduction to carry out this proof. A “commutation lemma” similar to the Church-Rosser property the existence of a type $S_1$ for $S \xrightarrow{\beta_T} S'_1$ and $S \xrightarrow{\beta_T} S_1$ where $S' \xrightarrow{\beta_T} S_1$ and $S \xrightarrow{\beta_T} S_1$, [lemma 3.8] is crucial for the induction. This immediately excludes the one-step reduction relation $\xrightarrow{\beta_T}$. Ordinary many-step reduction, $\xrightarrow{\beta_T}$, is another obvious choice; but it cannot be used in the case of R-App, since here we need to know the form of the reduce of the application, which cannot be recovered from $\xrightarrow{\beta_T}$. Other, even more deterministic, reduction strategy such as normalizing reduction, leftmost-innermost reduction, or complete development might work here, but these all seem to fail in the R-PRECEDE case. There, we have $S \xrightarrow{\beta_T} A \ S_1 \ldots S_n$ [or $\Gamma(A) \ S_1 \ldots S_n$, where $\Gamma(A) \ S_1$ can contain redices not present in $A \ S_1$]. For the induction to work in this case, we need to be able to ignore these new redices when reducing further from $\Gamma(A) \ S_1 \ldots S_n$, which we would not be free to do if we were using a more deterministic reduction strategy.}

31
Corollary: Cut-free derivations can be strengthened: we can put

We may assume that the derivations of the premises are cut-free. Moreover, by Corollary 6.4.9, we now show that the algorithm we developed informally in Section 6.1 is indeed a decision procedure for

Proposition: Termination of the algorithm:

By induction, we may assume that the derivations of the premises are cut-free. Moreover, by Corollary 6.4.9, we can put $U$ in normal form: $\Gamma \vdash S \leq U'$ and $\Gamma \vdash U' \leq T$. Corollary 6.4.10 allows these derivations to be strengthened: $\Gamma \vdash CS S \leq U'$ and $\Gamma \vdash CS U' \leq T$. The result now follows from Proposition 6.3.2.

6.6 The Algorithm

We now show that the algorithm we developed informally in Section 6.1 is indeed a decision procedure for the subtype relation.

The first thing we must verify is that this recursively defined procedure is really an algorithm — that it halts in finite time on all well-kind inputs.

Proposition: Termination of the algorithm:

Proof: We use the fact that the $\beta\eta$-reduction defined in Section 5 is strongly normalizing to define a simple termination ordering for the algorithm.

First, note that the recursive call in the third clause (the clause for promotion) is guaranteed to halt on the next step if $U$, the promotion of $S$, reduces to $\text{Top}(K)$ for some $K$. Thus, we need only consider the possibility of nontermination in the case where the promotion of $S$ is different from $\text{Top}$ — i.e., where the variable being promoted is a $\Gamma$-redex.

Let the rank of a well-kind type $V$ in a context $\Gamma$ be the pair $(r, s)$, where $r$ is the maximum length of a $\beta\eta$-reduction sequence starting from $V$ and $s$ is the number of characters in $V$. Let the rank of a well-kind subtyping statement $\Gamma \vdash S \leq T$ the pairwise sum of the ranks of $S$ and $T$. Order ranks lexicographically. Then the rank of every recursive call of $\text{check}$ in its definition is smaller than the rank of the input (using the observation that we need only consider “interesting promotions” in the third clause).

The algorithm $\text{check}$ does not include a case corresponding to the pointwise application rule $R$-$\text{App}$. But this rule can easily be shown to be inessential in strong, cut-free derivations.

Lemma: Eliminability of $R$-$\text{App}$:

Proof: Straightforward induction on derivations, using Lemma 6.3.1 for the $R$-$\text{App}$ case. In effect, each instance of the application rule is replaced by a sequence of instances of the promotion rule.

32
Finally, we verify that the algorithm defines the same relation as the original subtyping rules.

6.6.3 Theorem: The algorithm check is sound and complete for the original subtyping relation (on well-kinded types).

Proof: On well-kinded inputs, the original subtyping relation is equivalent to the reducing subtyping relation (6.2.7) restricted to strong, cut-free derivations (6.5.1) with no uses of R-App (6.6.2). Now, each of the rules in this restricted system begins by normalizing both sides of the conclusion. Nothing changes if we present the rules in a form where the conclusion is assumed to be in normal form, adding a single normalization step at the very end of the derivation and inserting a re-normalizing step at each premise that is not guaranteed to be in normal form when the conclusion is; indeed, there is just one of these: the last premise of R-Promote. We may now observe that proof search for derivations in the reformulated system is an essentially linear process: any given subtyping statement can match the conclusion of only one subtyping rule for which further search may be required. That is, a given statement may match R-Refl and/or R-Top and/or one of the remaining rules R-Promote, R-Arrow, R-All, and R-App. Since the premises of R-Refl and R-Top can be checked directly, the applicability of these two rules can be tested first. Using this strategy, no backtracking is required. Moreover, all the metavariables appearing in the premises the rules may be calculated from the conclusion: no guessing is required. The algorithm check implements this strategy.

\[ \square \]

7 Typing

In Section 6, we derived an algorithm for checking the subtyping relation by controlling the non-syntax-directed rules of transitivity and conversion. In this section, we carry out an analogous exercise for the typing relation, eliminating the rule of subsumption from the system defined in Section 2.5 and accounting for its effects by extending some of the other rules.

Compared to what we had to do for subtyping, this is actually a rather simple task. Indeed, the resulting algorithm strongly resembles standard algorithms for typechecking \( F_{\subseteq} \). The only essential difference comes from the fact that the promotion relation here must deal with application in addition to the promotion of type variables. As usual, we obtain the algorithm by analyzing the shapes of minimal types.

First, we check that the typing relation guarantees well-kindness of derivable statements:

7.1 Lemma: If \( \Gamma \vdash t : T \), then \( \Gamma \vdash T \in \ast \).

The minimal type of an expression is a type smaller or equal to all the other types of the expression. For the algorithm, we also need to talk about a term’s minimal types of certain specific shapes.

7.2 Definition [Minimal, arrow-minimal, and All-minimal types]:

1. A type \( S \) is minimal for a term \( s \) in a context \( \Gamma \) if \( \Gamma \vdash s : S \) and, for all \( T \) with \( \Gamma \vdash s : T \), we have \( \Gamma \vdash S \leq T \).

2. A type \( S_1 \rightarrow S_2 \) is arrow-minimal for \( s \) in \( \Gamma \) if \( \Gamma \vdash s : S_1 \rightarrow S_2 \) and, for all arrow-types \( T_1 \rightarrow T_2 \) with \( \Gamma \vdash T_1 \rightarrow T_2 \), we have \( \Gamma \vdash S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2 \).

3. A type \( \text{All}(A \leq S_1)S_2 \) is All-minimal for \( s \) in \( \Gamma \) if \( \Gamma \vdash s : \text{All}(A \leq S_1)S_2 \) and, for all All-types \( \text{All}(A \leq T_1)T_2 \) with \( \Gamma \vdash s : \text{All}(A \leq T_1)T_2 \), we have \( \Gamma \vdash \text{All}(A \leq S_1)S_2 \leq \text{All}(A \leq T_1)T_2 \).

7.3 Definition: Let \( T \) be well kinded in \( \Gamma \). The promote-normal form of \( T \) in \( \Gamma \) is

\[
\bar{t}_{\Gamma}^T = \begin{cases} \bar{t}_{\Gamma}^U \text{ if } T \vdash \bar{t}_{\Gamma}^U; \text{ or} \\ T \text{ if } T \text{ cannot be promoted.} \end{cases}
\]

We next show how All-minimal and arrow-minimal types of a term can be calculated from its minimal type.

33
7.4 Lemma:

1. Suppose $S$ and $T_1T_2$ are well kinded. If $\Gamma \vdash S \leq T_1T_2$, then $|_\Gamma S \equiv S_1T_2$ and $\Gamma \vdash S_1T_2 \leq T_1T_2$.

2. Suppose $S$ and $A\Gamma\Delta T_2$ are well kinded. If $\Gamma \vdash S \leq A\Gamma\Delta T_2$, then $|_\Gamma S \equiv A\Gamma\Delta S_1T_2$ and $\Gamma \vdash A\Gamma\Delta S_1T_2 \leq A\Gamma\Delta T_1T_2$.

Proof:

1. On well-kindness inputs, the original subtyping relation is equivalent to the reducing subtyping relation (6.2.7) restricted to strong, cut-free derivations (6.5.1). Proceed by induction on a derivation of $\Gamma \vdash S \leq T_1T_2$.

2. Similar. \qed

7.5 Corollary:

1. If $S$ is minimal for $s$ in $\Gamma$ and $|_\Gamma S$ has the form $S_1T_2$, then $S_1T_2$ is arrow-minimal for $s$ in $\Gamma$.

2. If $S$ is minimal for $s$ in $\Gamma$ and $|_\Gamma S$ has the form $A\Gamma\Delta S_2$, then $A\Gamma\Delta S_1T_2$ is All-minimal for $s$ in $\Gamma$.

The typing algorithm can now be obtained directly from the original typing relation by removing T-SUBSUMPTION — in effect, restricting the set of types derivable for a well-typed term to one of its minimal types — and generalizing the application and type application rules to compensate for this restriction in their premises. We use $\vdash_A$ to distinguish the typing algorithm from the original typing relation.

7.6 Definition [Typechecking algorithm]:

\[
\begin{align*}
&\frac{}{\Gamma \vdash \text{ok}} \quad \text{(A-VAR)} \\
&\frac{}{\Gamma, x: T_1 \vdash_A e \in T_2} \quad \text{(A-ARROW-I)} \\
&\frac{\Gamma \vdash_A \text{fun}(x: T_1)e \in T_1T_2}{\Gamma \vdash_A s \in S \quad \Gamma \vdash_A t \in T} \quad \text{(A-ARROW-E)} \\
&\frac{\Gamma \vdash_A s \in S \quad \Gamma \vdash_A t \in T \quad \Gamma \vdash T \leq T_1}{\Gamma \vdash_A \text{fun}(s: T_1)e \in A\Gamma\Delta T_2} \quad \text{(A-ALL-I)} \\
&\frac{\Gamma \vdash_A \text{fun}(A\Gamma\Delta T_1)e \in A\Gamma\Delta T_2}{\Gamma \vdash_A t \in T \quad \Gamma \vdash S \in K \quad \Gamma \vdash S \leq T_1}{\Gamma \vdash_A tS \in [S/A]T_2} \quad \text{(A-ALL-E)}
\end{align*}
\]

The termination of this algorithm is straightforward, given the decidability of kinding, the termination of the subroutine for checking subtyping, and the strong normalization of $\beta\xi$-reduction, which guarantees that $|_\Gamma T$ can be calculated in finite time whenever $T$ is well kinded.

7.7 Fact:

1a. If $\vdash \Gamma \text{ok}$, then $\Gamma(x)$ is a minimal type of $x$ in $\Gamma$.

1b. If $\not\vdash \Gamma \text{ok}$, then $x$ has no type in $\Gamma$.

2a. If $T_2$ is a minimal type of $e$ in $\Gamma$, $x: T_1$, then $T_1 \rightarrow T_2$ is a minimal type of $\text{fun}(x: T_1)e$ in $\Gamma$.

2b. If $e$ has no type in $\Gamma$, $x: T_1$, then $\text{fun}(x: T_1)e$ has no type in $\Gamma$. 

34
3a. If $S$ is a minimal type for $s$ in $\Gamma$ and $\vdash_{\Gamma} S \Rightarrow T_1 \rightarrow T_2$ and $T$ is a minimal type for $t$ in $\Gamma$ and $\Gamma \vdash T \leq T_1$, then $T_2$ is a minimal type for $s$ $t$ in $\Gamma$.

3b. If $s$ or $t$ has no type in $\Gamma$, or if $S$ is a minimal type for $s$ in $\Gamma$ and $T$ is a minimal type for $t$ in $\Gamma$ but $\vdash_{\Gamma} S \not\Rightarrow T_1 \rightarrow T_2$, or if $\vdash_{\Gamma} S \Rightarrow T_1 \rightarrow T_2$ but $\Gamma \not\vdash T \leq T_1$, then $s$ $t$ has no type in $\Gamma$.

4a. If $T_2$ is a minimal type of $e$ in $\Gamma$, $A \leq T_1$, then $\text{All}(A \leq T_1) T_2$ is a minimal type of $\text{fun}(A \leq T_1) e$ in $\Gamma$.

4b. If $e$ has no type in $\Gamma$, $A \leq T_1$, then $\text{fun}(A \leq T_1) e$ has no type in $\Gamma$.

5a. If $S$ is a minimal type for $s$ in $\Gamma$ and $\vdash_{\Gamma} S = \text{All}(A \leq T_1) T_2$ and $\Gamma \vdash U \leq T_1$, then $[U/A]T_2$ is a minimal type for $s$ $U$ in $\Gamma$.

5b. If $s$ has no type in $\Gamma$ or if $S$ is a minimal type for $s$ in $\Gamma$ but $\vdash_{\Gamma} S \not= \text{All}(A \leq T_1) T_2$ or if $\vdash_{\Gamma} S = \text{All}(A \leq T_1) T_2$ but $\Gamma \not\vdash U \leq T_1$, then $s$ $U$ has no type in $\Gamma$.

7.8 Theorem [Soundness and completeness]:

1. If $\Gamma \vdash_A t \in T$ then $\Gamma \vdash T \in \star$ and $\Gamma \vdash t \in T$.

2. If $\Gamma \vdash s \in T$ then $\Gamma \vdash T \in \star$ and $\Gamma \vdash_A s \in S$, where $S$ is minimal for $s$ in $\Gamma$.

Proof: By induction, using the previous facts. □

This brings us to our final result:

7.9 Corollary [Decidability of $F^\leq_{\leq}$ typing]: The original typing relation $\Gamma \vdash t \in T$ is decidable.

Proof: To check whether a statement $\Gamma \vdash t \in T$ is derivable, first check that $T$ is well kinded, then calculate the minimal type $S$ of $t$ in $\Gamma$ using the algorithm above and use the subtyping algorithm check to verify that $\Gamma \vdash S \leq T$. □

Acknowledgements

This research was supported by the Specification and Verification of Distributed Systems project, funded by the Deutsche Forschungsgemeinschaft, Sonderforschungsbereich 182 "Multiprozessor- und Netzwerkkonfigurationen" with assistance from the DAAD (Deutscher Akademischer Austauschdienst) [Steffen], by the British Science and Engineering Research Council [Pierce], and by and the British Council [Steffen and Pierce]. Discussions with Ulf Becker, Luca Cardelli, Adriana Compagnoni, Randy Pollack, Terry Stroup, and David Turner helped deepen our understanding of $F^\leq_{\leq}$. David Aspinall and Adriana Compagnoni gave us useful comments on a late draft. Giorgio Ghelli pointed out a bug in an earlier version of the proof, which Adriana Compagnoni helped correct.

References


